# PENGEMBANGAN APLIKASI *MOBILE LOCATION BASED SERVICE* BERBASIS ANDROID UNTUK PENCARIAN LOKASI RUMAH SAKIT DI KOTA MALANG BERDASARKAN ASURANSI KESEHATAN DENGAN METODE *AGILE SYSTEM DEVELOPMENT*

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sabil Prihastomo Seputro
NIM: 145150201111052

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

# PENGESAHAN

PENGEMBANGAN APLIKASI *MOBILE LOCATION BASED SERVICE* BERBASIS ANDROID UNTUK PENCARIAN LOKASI RUMAH SAKIT DI KOTA MALANG BERDASARKAN ASURANSI KESEHATAN DENGAN METODE *AGILE SYSTEM DEVELOPMENT*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sabil Prihastomo Seputro
NIM: 145150201111052

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Dr.Eng. Herman Tolle, S.T., M.T. | Komang Candra Brata, S.Kom., M.T., M.Sc. |
| NIP: 19740823 200012 1 001 | NIK. 2016078907111001 |

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 2003121 1 001

## PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018


Sabil Prihastomo Seputro

NIM: 145150201111052

# KATA PENGANTAR

Assalamualaikum Wr. Wb. Alhamdulillah, puji syukur kehadirat Allah subhanahu wa ta'ala atas segala limpahan rahmat-Nya sehingga skripsi ini dapat dikerjakan dengan lancar dan terselesaikan dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Strata satu (S1) Sarjana Komputer dari jurusan Teknik Informatika, Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya.

Dalam pengerjaan skripsi ini telah melibatkan banyak pihak yang turut membantu dalam proses pengerjaan, sehingga dapat diselesaikan dengan lancar. Pada kesempatan ini penulis ingin menyampaikan rasa terima kasih sebesar-besarnya kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universtitas Brawijaya.
2. Bapak Tri Astoto Kurniawan , S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
3. Bapak Agus Wahyu Widodo , S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya
4. Bapak Dr.Eng. Herman Tolle, S.T., M.T. selaku dosen pembimbing pertama yang telah memberikan bimbingan, pengarahan, dan saran dalam penyusunan skripsi ini.
5. Bapak Komang Candra Brata, S.Kom., M.T., M.Sc. selaku dosen pembimbing kedua yang telah memberikan bimbingan, pengarahan, dan saran dalam penyusunan skripsi ini.
6. Bapak, Ibu dosen serta segenap staff dan karyawan Jurusan Teknik Informatika yang telah membantu dalam menyelesaikan skripsi ini.
7. Supriyono Wignyo Seputro, Lilyk Happy Priyamsari, Mareta Romadhona Primaningtyas, Nur Wira Prihastami Seputri, selaku keluarga tercinta yang tiada henti memberikan semangat, dukungan dan doa yang tiada henti.
8. Akbar Noegrahing S.M, Luthfi Abdul Basith, Indra Fahrizal, Yusuf Wicaksono, Akyas Trenggono, dan teman-teman seperjuangan di PPM angkatan 2014.
9. Kawan-kawan yang tidak hentinya memberikan canda dan tawa : Kediaman, PPM Al-Kautsar, PPM Baitul Jannah, PPM Nur Muhammad dan seluruh kawan-kawan yang tidak tersebut.
10. Teman-teman seperjuangan dari pembimbing Dr.Eng. Herman Tolle, S.T., M.T.
11. Teman-teman seperjuangan Teknik Informatika 2014.
12. Semua pihak yang telah memberikan bantuan baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini.

Penulis menyadari bahwa masih terdapat kekurangan pada skripsi yang disusun oleh penulis, sehingga mohon maaf apabila terdapat penulisan yang kurang tepat. Semoga skripsi ini dapat bermanfaat bagi khalayak umum guna mendukung kemajuan ilmu terkhusus pada bidang pendidikan. Wassalamualaikum Wr. Wb.

Malang, 2 Agustus 2018

Penulis

sabil_ps@yahoo.com

# ABSTRAK

Kota Malang merupakan salah satu kota besar di Jawa Timur dan perkembangan kesehatannya mulai berkembang setiap tahunnya, dan salah satu layanan kesehatan yang banyak tersedia adalah rumah sakit. Sekarang masyarakat Kota Malang dapat berobat ke rumah sakit dengan mudah dengan adanya asuransi kesehatan. Tetapi dengan adanya berbagai macam asuransi kesehatan menimbulkan berbagai masalah. Masyarakat yang memiliki salah satu jenis asuransi kesehatan kebingungan dalam mencari rumah sakit yang menerima asuransi kesehatan yang dia miliki karena tidak semua rumah sakit menerima asuransi kesehatan. Masyarakat yang sedang di posisi gawat darurat dan membutuhkan penanganan dokter secara cepat harus menemukan rute ke rumah sakit terdekat. Bila rumah sakit yang dituju tidak menerima asuransi kesehatan yang dimiliki oleh masyarakat dan terlambat dalam penanganannya, maka akan berakibat fatal bahkan dapat berujung kematian. Untuk mengurangi dampak tersebut, masyarakat harus dapat mencari rumah sakit berdasarkan asuransi kesehatan dan *routing* dari lokasi pengguna ke lokasi rumah sakit, dengan membuat aplikasi menggunakan *GPS* & *API Google Maps* dengan metode *Agile System Development* yang diimplementasikan di Android. Pembuatan aplikasi *Hospital Locator* bertujuan untuk mengetahui kinerja dari setiap fungsional sistem, maka akan diuji melalui *validity testing* serta dilakukan pengujian *usability testing* berupa *usability score* dan skala *Likert* untuk mengetahui kualitas dari sistem. Hasil dari *validity testing* didapatkan fungsi dari sistem telah berjalan dengan baik dengan tingkat keberhasilan 100%. Hasil dari *usability testing* sistem mendapatkan skor 92,91% yang masuk dalam kategori A yang mana pengguna dapat menggunakan aplikasi dengan mudah dan merasa terbantu.

**Kata kunci:** lokasi, *routing, GPS, API, Google Maps, Android, Validity Testing, Usability Testing*

# ABSTRACT

Malang City is one of the big cities in East Java and its health development began to develop every year, and one of the health service that widely available is hospital. Now, people in Malang City can get treatment to the hospital easily with the existence of health insurance. But with the existence of various kinds of health insurance causes various troubles. People who own one kind of the health insurance confused when they want to search the hospital that received their health insurance because not all of the hospital will received all health insurance. People who were in emergency situation and require doctor attention immediately must find the nearest route to the hospital. If the targeted hospital does not receive their health insurance and the handling is late, it will be fatal and may even lead to death. To decrease those impact, people must able to search the hospital based on health insurance and routing from user location to the hospital location, with making an application that using GPS & API Google Maps with Agile System Development method that implemented on Android. The creation of Hospital Locator application intends to know the performance of each functional of the system, then it will be tested through validity testing as well as be testing with usability testing in the form of usability score and Likert scale to know the quality of the system. The result of the validity testing obtained that the functional of the system has been running well with 100% success rate. The result of the usability testing, the system gets score 92,91% which can be categorized as an A and user can operate the system easily and feel helpful about it.

**Keywords:** *location*, *routing, GPS, API, Google Maps, Android, Validity Testing, Usability Testing*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR KODE

# DAFTAR LAMPIRAN

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Kesehatan merupakan hal yang sangat penting agar manusia dapat bertahan hidup dan melakukan aktivitas. Pentingnya kesehatan ini mendorong pemerintah untuk mendirikan layanan kesehatan, agar masyarakat dapat mengakses kebutuhan kesehatan. Layanan kesehatan adalah salah satu jenis layanan publik yang merupakan ujung tombak dalam pembangunan kesehatan masyarakat (Nafsiah, 2000).

Rumah sakit adalah layanan kesehatan untuk melakukan upaya meningkatkan kesehatan, mencegah dan menyembuhkan penyakit, serta memulihkan kesehatan. Pengelolaan unit usaha rumah sakit memiliki keunikan tersendiri karena selain sebagai unit bisnis, baik itu rumah sakit milik pemerintah ataupun milik swasta. Usaha rumah sakit juga memiliki misi sosial yang berperan penting dalam hal kesehatan yang merupakan suatu unit usaha jasa yang memberikan jasa pelayanan sosial dibidang medis klinis yang diperkuat oleh Departemen Kesehatan RI (1999). Salah satu layanan kesehatan publik di salah satu kota besar di Jawa Timur seperti Malang adalah rumah sakit.

Hakikat dasar dari rumah sakit adalah pemenuhan kebutuha dari tuntutan pengguna yang mengharapkan penyelesaian masalah kesehatannya dengan berbagai akses untuk mendapatkannya. Masyarakat memandang bahwa hanya rumah sakit yang mampu memberikan pelayanan medis terbaik sebagai upaya penyembuhan dan pemulihan atas rasa sakit yang dideritanya, untuk itu rumah sakit pada umumnya harus memberikan pelayanan yang bermutu sesuai dengan standar yang ditetapkan dan dapat menjangkau seluruh masyarakat (Departemen Kesehatan RI, 1999).

Sekarang ini masyarakat dapat berobat ke rumah sakit dengan menggunakan bermacam-macam asuransi kesehatan yang tersedia, baik itu asuransi kesehatan yang berasal dari pemerintah ataupun swasta. Dengan adanya berbagai macam asuransi kesehatan yang tersedia seperti *AdMedika, ABDA, Allianz, BPJS, Mandiri InHealth, Prudential*, dll menimbulkan berbagai masalah. Masyarakat yang memiliki salah satu jenis asuransi kesehatan kebingungan dalam mencari rumah sakit yang menerima asuransi kesehatan yang dia miliki karena tidak semua rumah sakit menerima asuransi kesehatan yang dimiliki oleh masyarakat tersebut. Masyarakat yang saat posisi gawat darurat dan membutuhkan penanganan dokter secara cepat harus menemukan rute ke rumah sakit terdekat. Bila rumah sakit yang dituju tidak menerima asuransi kesehatan yang masyarakat punya dan terlambat dalam penanganannya, maka akan berakibat fatal bahkan dapat berujung kematian (Thabrany, 2001).

Berdasarkan uraian tersebut, diharapkan permasalahan masyarakat di Kota Malang dapat diselesaikan dengan melakukan pencarian rumah sakit berdasarkan preferensi asuransi kesehatan menggunakan *GPS & API Google Maps* dengan metode *Agile System Development* yang diimplementasikan di Android. Android

digunakan dalam penelitian ini karena banyaknya pengguna Android di Indonesia yang diperkuat oleh Brodkin (2012). *GPS & API Google Maps* digunakan untuk merouting dari lokasi pengguna ke lokasi rumah sakit. Sedangkan *Agile System Development* digunakan karena bila sewaktu-waktu ada asuransi kesehatan atau rumah sakit yang baru tidak perlu diadakan perombakan sistem secara besar-besaran karena adanya *Iterative* dan *Incremental*. Kemudian karena akan banyaknya pengguna yang akan memakai aplikasi ini *Agile System Development* sangat baik digunakan karena adanya keterlibatan pengguna secara langsung dalam pembuatannya dan dapat merubah requirement sewaktu-waktu yang mana mungkin pengguna membutuhkan fitur tersebut. Diharapkan dengan mendapatkan data asuransi yang diterima oleh rumah sakit di Kota Malang akan lebih mudah mencari rumah sakit yang menerima asuransi kesehatannya. Dimana dalam proses pembuatan akan mencari tahu bagaimana analisis kebutuhan, perancangan sistem dan implementasi dari perangkat lunak, juga diperlukan sebuah pengujian apakah perangkat lunak tersebut sudah sesuai perancangan dengan *validity testing* dan dapat diterima oleh pengguna dalam sebuah *usability testing*.

## 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang tersebut, maka dalam pelaksanaan penelitian ini, dirumuskan beberapa rumusan masalah sebagai berikut:

1. Bagaimana hasil analisis kebutuhan, perancangan sistem dan implementasi Aplikasi *Mobile Location Based Service* berbasis Android untuk pencarian lokasi rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan yang dimiliki pengguna?
2. Bagaimana hasil *validity testing* dan *usability testing* dari Aplikasi *Mobile Location Based Service* berbasis Android untuk pencarian lokasi rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan yang dimiliki pengguna?

## 1.3 Tujuan

Berdasarkan uraian rumusan masalah tersebut, maka tujuan yang hendak dicapai dalam penelitian ini adalah:

1. Untuk mengetahui hasil analisis kebutuhan, perancangan sistem dan implementasi Aplikasi *Mobile Location Based Service* berbasis Android untuk pencarian lokasi rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan yang dimiliki pengguna.
2. Untuk mengetahui hasil *validity testing* dan *usability testing* Aplikasi *Mobile Location Based Service* berbasis Android untuk pencarian lokasi rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan yang dimiliki pengguna.

## 1.4 Manfaat

Manfaat yang dapat didapatkan dari penelitian ini adalah:

1. Terciptanya suatu aplikasi yang mempermudah dalam mencari rumah sakit yang terdekat berdasarkan preferensi asuransi kesehatan yang yang dimiliki pengguna.

## 1.5 Batasan Masalah

Untuk mengarahkan penelitian agar sesuai dengan permasalahan yang telah ditentukan, maka diberikan Batasan masalah sebagai berikut:

1. Studi kasus yang digunakan dalam penelitian ini di kota Malang.
2. Aplikasi yang digunakan untuk pengembangan adalah Android *Studio* 2.3.3.
3. Android minimal versi *Lollipop* - Android 5.1.
4. Rumah sakit yang digunakan dalam penelitian ini adalah RSB. Mardi Waloeja, RSB. Permata Hati Malang, RSI. Aisyiyah, RSI. Malang UNISMA, RSIA. Galeri Candra, RSIA. Ganesha Medika, RSIA. Husada Bunda, RSIA. Melati Husada, RSIA. Mutiara Bunda Malang, RSIA. Puri Bunda Malang, RSIA. Puri Medika Malang, RSIA. Refa Husada, RST. Dr. Soepraoen, RSU. Bhakti Bunda Malang, RSU. Hermina Tangkubanprahu, RSU. Lavalette, RSU. Panti Nirmala, RSU.Panti Waluya, RSU.Permata Bunda Malang, RSU.Persada, RSU. Universitas Brawijaya, RSU. Universitas Muhammadiyah Malang, RSUD. DR. Saiful Anwar.
5. Asuransi kesehatan yang digunakan dalam penelitian ini adalah *AA Internasional, ABDA, Adira, AdMedika, AIA, Allianz, Astra Life, AVIVA, AXA, Bintang, BPJS, BRIngin Life, CAR, CHUBB, CIGNA, Common Wealth, EQUITY, FWD, Garda Medika, Great Eastern, Hanwha, I'm Care 177, Jagadiri, Jasindo, Jiwasraya, Kresna, Lippo, MAG, Mandiri InHealth, Medilum, Mega, MNC Life, NAYAKA, Pacific Cross, Pan Pacific, Panin, Prudential, Reliance, Sinarmas, SMP, SOS, Takaful, TMS.*
6. Implementasi peta menggunakan *API Google Maps* Android.

## 1.6 Sistematika Pembahasan

Laporan penelitian ini ditulis dalam garis besar laporan yang terdiri dari beberapa bagian/bab sebagai berikut:

### BAB 1 PENDAHULUAN

Memuat latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian / ruang lingkup, sistematika pembahasan / laporan.

## BAB 2 LANDASAN KEPUSTAKAAN

Berisi tentang teori-teori yang mendasari dan mendukung pembuatan Aplikasi *Mobile Location Based Service* berbasis Android untuk pencarian lokasi rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan yang dimiliki.

## BAB 3 METEDOLOGI PENELITIAN

Memuat metode penelitian atau langkah-langkah penelitian yang dilakukan. Langkah-langkah penelitian pada bab ini adalah studi pustaka, analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan penutup dari laporan.

## BAB 4 ANALISIS KEBUTUHAN

Bab ini menguraikan tentang proses analisis kebutuhan. Menjelaskan proses dan hasil analisis domain serta penggalian kebutuhan-kebutuhan yang menjadi dasar pembangunan sistem pencarian rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan.

## BAB 5 PERANCANGAN SISTEM

Bab ini membahas proses perancangan sistem pencarian rumah sakit di kota Malang berdasarkan hasil analisis kebutuhan. Pada bagian ini dipaparkan hasil rancangan arsitektur, deskripsi tiap komponen penyusun, serta model data yang ada dalam sistem.

## BAB 6 IMPLEMENTASI

Bab ini akan membahas mengenai hasil proses implementasi rancangan sistem pencarian rumah sakit di kota Malang yang diperoleh melalui tahapan perancangan sistem. Dalam bagian ini dipaparkan penjelasan mengenai bahasa pemrograman yang digunakan, metode yang digunakan, serta prosedur-prosedur yang dilakukan pada pembuatan perangkat lunak.

## BAB 7 PENGUJIAN

Bab ini menguraikan strategi pengujian, data hasil, dan analisis hasil dari kegiatan pengujian sistem pencarian rumah sakit di kota Malang yang telah dirancang dan diimplementasikan sebelumnya kemudian dilakukan evaluasi dari pengujian tersebut apakah aplikasi sudah sesuai.

## BAB 8 PENUTUP

Bab ini menguraikan tentang ringkasan dari keseluruhan proses pengembangan sistem dan memaparkan hasil pencapaian dari penelitian pengembangan ini. Selain itu, pada bagian ini akan dicantumkan pesan dan saran-saran serta pengembangan lanjut yang dapat dilakukan terhadap penelitian.

# BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini berisi uraian dan pembahasan tentang teori, konsep, model, metode, dan sistem dari literatur yang berkaitan dengan tema, masalah, atau pertanyaan penelitian yang diperkuat oleh Mahsun (2011). Seluruh uraian dan pembahasan ini digunakan sebagai dasar dalam melaksanakan pengembangan aplikasi *mobile location based service* berbasis Android untuk pencarian lokasi rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan dengan metode *agile system development*

## 2.1 Tinjauan Pustaka

Penelitian terdahulu yang menjadi bahan kajian pustaka utama dalam penelitian ini adalah penelitian berjudul "Rancangbangun Pencarian Lokasi Rumah Sakit dan Puskesmas di Wilayah Tegal Berbasis Android" ( Hartantyo, 2014).

Dalam penelitian tersebut pengembangan sistem berusaha mengembangkan sebuah sistem untuk mempermudah masyarakat dalam mencari sebuah informasi tentang lokasi rumah sakit dan puskesmas di wilayah Tegal. Akan tetapi, berdasarkan penelusuran yang dilakukan, pengembangan sistem ini belum di buat di Kota Malang yang mana merupakan salah satu kota besar di Jawa Timur. Dengan dasar tersebut, penelitian ini akan mengembangkan sebuah sistem tentang lokasi rumah sakit yang sama tetapi di wilayah Kota Malang.

## 2.2 Asuransi Kesehatan

Asuransi kesehatan merupakan bentuk kerja sama untuk menghindari atau meminimalkan risiko terjadinya kematian. Ketika seorang menderita suatu penyakit atau mengalami kecelakaan sering muncul adanya kerugian finansial (terganggunya penghasilan atau pengeluaran untuk pengobatan). Asuransi kesehatan didesain untuk menyediakan santunan yang digunakan untuk membayar sebagian dari kerugian finasial tersebut. Asuransi kesehatan perawatan rumah sakit adalah asuransi kesehatan yang memberikan santunan kesehatan kepada seseorang (tertanggung) berupa sejumlah uang untuk biaya pengobatan dan perawatan bila diluar kehendaknya ia diserang penyakit. Sebagai imbalan atas santunan kesehatan yang diberikan oleh penanggung maka tertanggung membayar premi kepada penanggung, pada umumnya dibayar secara berkala, misal bulanan, triwulan, semester atau tahunan. Polis yang digunakan dapat berupa polis seumur hidup (*wholelife*) atau polis berjangka (*term-life*)(Thabrany, 2001).

Masalah yang terjadi di kota Malang banyak masyarakat yang memiliki asuransi kesehatan seperti *AdMedika, ABDA, Allianz, BPJS, Mandiri InHealth, Prudential*, dll yang menimbulkan masalah baru di kota Malang (Prawoto, Agus, 1995). Masyarakat yang memiliki salah satu jenis asuransi kesehatan kebingungan dalam mencari rumah sakit yang menerima asuransi kesehatan yang dia miliki karena tidak semua rumah sakit menerima asuransi kesehatan yang dimiliki oleh

masyarakat tersebut. Masyarakat yang saat posisi gawat darurat dan membutuhkan penanganan dokter secara cepat harus menemukan rute ke rumah sakit terdekat. Bila rumah sakit yang dituju tidak menerima asuransi kesehatan yang masyarakat punya dan terlambat dalam penanganannya, maka akan berakibat fatal bahkan dapat berujung kematian.

## 2.3 *Location Based* Service (LBS)

*Location Based Service* (LBS) adalah layanan untuk menyediakan informasi yang telah ada dibuat, dikompilasi, dipilih, atau disaring dengan mempertimbangkan lokasi saat ini pengguna atau orang lain atau benda bergerak. Mereka juga dapat muncul bersamaan dengan layanan konvensional seperti telepon dan fitur nilai tambah terkait, misalnya untuk direalisasikan perutean panggilan berbasis lokasi atau pengisian berbasis lokasi. Daya tarik hasil *Location Based Service* adalah dari fakta bahwa peserta mereka tidak perlu memasukkan informasi lokasi secara manual, tetapi mereka secara otomatis ditunjuk dan dilacak(Wiley, 2005).

Karena itu, teknologi utamanya adalah *positioning*, dimana berbagai metode ada yang berbeda satu sama lain dalam sejumlah kualitas parameter dan keadaan lainnya. Begitu informasi lokasi diturunkan, perlu diproses dalam beberapa cara, termasuk transformasi ke dalam format ruang lain sistem referensi, korelasinya dengan informasi lokasi atau konten geografis lainnya, generasi peta, atau perhitungan petunjuk navigasi, untuk beberapa nama saja (Putro, A., Tolle, H., & Kharisma, A, 2017).

*Location Based Service* dalam penelitian ini dipakai sebagai dasar pembuatan aplikasi yang mana digunakan dalam pencarian lokasi rumah sakit yang berada di kota Malang. *Location Based Service* akan menunjukan lokasi dari pengguna dan lokasi rumah sakit.

## 2.4 *API Google Maps*

*API Google Maps* adalah fungsi-fungsi pemrograman yang disediakan oleh *Google maps* agar *Google* maps dapat di integrasikan kedalam *Web* atau aplikasi yang sedang buat. Contoh sederhanya misalkan anda ingin membuat Sistem informasi Geografis kampus di Jogja, dengan memanfaatkan *API Google Maps* anda dapat membuat *GIS* tanpa perlu memikirkan Peta Jogja, anda tinggal pake *Google maps* dan memanggil fungsi fungsi yang dibutuhkan seperti menampilkan peta, menempatkan marker dan sabagainya. *Google Place API* ini adalah *API* wajib yang harus dikuasai untuk membuat aplikasi *Location based service*. (Ingraham, 2012).

*GPS & API Google Maps* digunakan untuk menunjukan lokasi dari pengguna dan lokasi rumah sakit dan merouting dari lokasi pengguna ke lokasi rumah sakit.

## 2.5 Android

Android adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tabelt. Android awalnya dikembangkan oleh Android*, Inc*., dengan dukungan finansial dari *Google*, yang kemudian membelinya pada tahun 2005(Elgin, 2005).

Android dikembangkan secara pribadi oleh *Google* sampai perubahan terbaru dan pembaruan siap untuk dirilis, dan informasi mengenai kode sumber juga mulai diungkapkan kepada publik. Kode sumber ini hanya akan berjalan tanpa modifikasi pada perangkat tertentu, biasanya pada seri *Nexus*. Ada binari tersendiri yang disediakan oleh produsen agar Android dapat beroperasi(McCann, 2012).

Android digunakan dalam penelitian ini karena banyaknya pengguna Android di Indonesia yang diperkuat oleh Brodkin (2012). Android minimal yang dapat digunakan untuk menjalankan aplikasi adalah versi *Lollipop* - Android 5.1

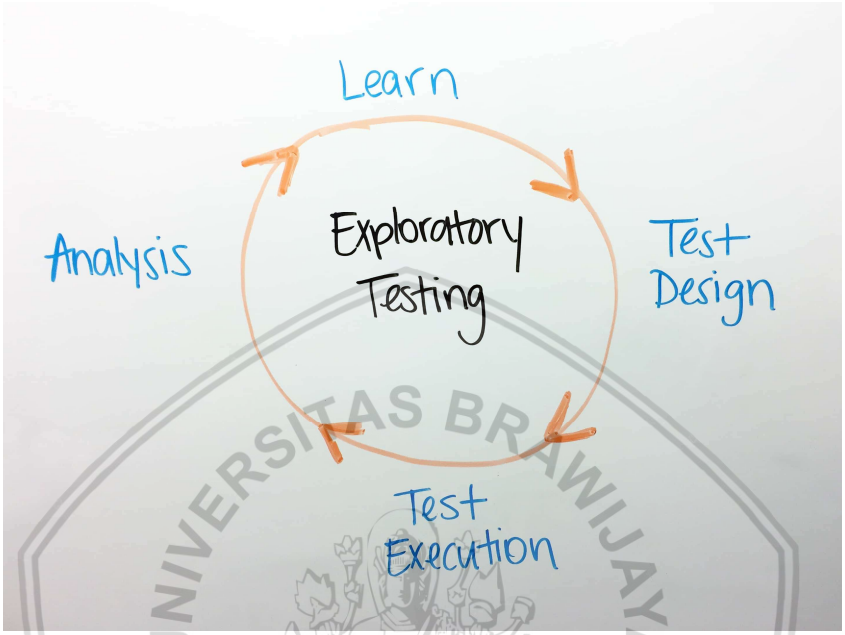
## 2.6 *Agile System Development*

*Agile Development Methods* adalah metodologi pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang sama atau pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. *Agile system development* dengan *exploratory testing* memiliki empat fase utama yaitu *analysis, learn, test design dan test execution* yang dapat dilihat pada Gambar 2.1 yang mana merupakan salah satu dari metodologi pengembangan perangkat lunak yang digunakan dalam pengembangan perangkat lunak. *Agile system development* dengan *exploratory testing* yang sebenarnya merupakan jenis pengujian fungsional tetapi penting dalam lingkungan *Agile*. Dalam hal ini, penguji tidak mengikuti langkah-langkah pengujian, melainkan menggunakan perangkat lunak dengan cara standar atau cerdas untuk mencoba memecahkannya. Penguji ini tentang mengembangkan pengujian terbaik berdasarkan setiap perangkat lunak unik. Karena pendekatan tanpa naskah, *exploratory testing* sering meniru bagaimana pengguna akan berinteraksi dengan perangkat lunak dalam kehidupan nyata (Ambler, 2012).

Apa bedanya dengan *Waterfall Testing* standar? *Exploratory testing* sebenarnya dapat dilakukan di lingkungan *Waterfall Testing* dan *Agile*, tetapi integrasi ketat antara penguji dan pengembang di lingkungan *Agile* membantu mengurangi hambatan yang mungkin muncul saat menjalankan *exploratory testing* di lingkungan *Waterfall*. *Exploratory testing* dapat membantu mengurangi waktu pengujian yang dihabiskan, menemukan lebih banyak *errors* dan meningkatkan cakupan kode. Akibatnya, pengujian eksploratif paling sesuai untuk yang berada di bawah batasan waktu yang perlu membantu mengidentifikasi jenis fungsionalitas terbaik untuk dijalankan (terutama dalam kasus di mana tidak ada spesifikasi dari pengembang) (Ambler, 2012).

*Agile System Development* digunakan dalam penelitian ini karena bila sewaktu-waktu ada asuransi kesehatan dan rumah sakit yang baru di Kota Malang tidak perlu merobak seluruh sistem karena metode pengembangan perangkat

lunak yang iteratif, selalu mengalami perubahan, dan evolusioner sangat dipertimbangkan sangat sesuai untuk aplikasi ini kedepannya karena akan ada evaluasi aplikasi kedepannya demi kenyamanan, efektifitas dan efisiensi pengguna dalam pemakaian aplikasi kedepannya.




**Gambar 2.1 *Agile Exploratory Testing***

**Sumber: https://d1f5pmhur9pirz.cloudfront.net/wp-content/uploads/2017/08/exp.jpg**

## 2.7 Konsep Rekayasa Aplikasi Perangkat Bergerak

Rekayasa perangkat lunak adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembanganan perangkat lunak dan manajemen kualitas. *IEEE Computer Society* mendefinisikan rekayasa perangkat lunak sebagai penerapan suatu pendekatan yang sistematis, disiplin dan terkuantifikasi atas pengembangan, penggunaan dan pemeliharaan perangkat lunak, serta studi atas pendekatan-pendekatan ini, yaitu penerapan pendekatan *engineering* atas perangkat lunak(Abran & Moore, 2004).

Pada fase perancangan, kebutuhan yang telah didapatkan dimodelkan menjadi diagram. Diagram bertujuan untuk memudahkan dokumentasi dari aplikasi yang sedang dikembangkan. Diagram yang perlu dibuat adalah sebagai berikut:

1. *Use case diagram* digunakan untuk menggambarkan cara kerja sistem secara umum dan didalam penelitian ini dengan aktor yaitu pengguna yang memiliki asuransi kesehatan tertentu (Gemino & Parker, 2009).

**Tabel 2.1 Simbol-simbol di dalam *Use Case Diagram***

| No | Nama Simbol | Deskripsi |
|---|---|---|
| 1 | *Use Case*<br>Nama *use case* | Merupakan fungsionalitas yang memiliki bentuk sistem sebagai unit - unit yang saling bertukar pesan |
| 2 | Aktor<br>  | Merupakan representasi aktor dalam aplikasi |
| 3 | Asosiasi<br> | Merupakan representasi komunikasi antar aktor atau *use case* |
| 4 | Ekstensi/*extend*<br><<extend>><br>  | Merupakan relasi *use case* tambahan |
| 5 | Generalisasi<br>  | Merupakan hubungan generalisasi |
| 6 | *Include*<br><<include>><br>  | Merupakan *use case* tambahan yang merupakan syarat untuk dijalankanya *use case* ini |

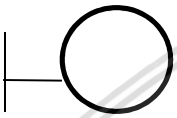
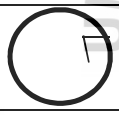
2. Pembuatan *use case scenario* digunakan untuk menggambarkan alur kerja sistem yang dijalankan. Di penelitian ini sistem dijalankan menjadi tiga belas *use case scenario* yang mana menggambarkan pengguna dari awal sistem dimulai sampai akhir sistem dijalankan( Cockburn, 2002).

**Tabel 2.2 Simbol-simbol di dalam *Use Case Scenario***

| *Flow of events* yaitu aliran - aliran *event* | |
|---|---|
| *Actors* | Aktor yang bersangkutan |
| *Objective* | Merupakan tujuan yang akan dicapai |
| *Pre-condition* | Berisi kondisi - kondisi sebelumnya |
| *Main flow* | Berisi aliran utama |
| *Alternative flows* | Berisi aliran - aliran alternatif |
| *Post-condition* | Berisi kondisi - kondisi setelahnya |

3. *Sequence diagram* digunakan untuk menggambarkan cara kerja aktor berinteraksi dengan sistem. Di penilitian ini menggambarkan aktor berinteraksi dengan sistem peraksi yang dilakukan oleh aktor(OMG, 2011).

**Tabel 2.3 Simbol-simbol di dalam *Sequence Diagram***

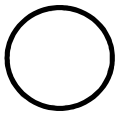
| No | Nama symbol | Definisi |
|---|---|---|
| 1 | Aktor  | Merupakan representasi aktor dalam aplikasi |
| 2 |  | Merupakan *boundary class* yang menghubungkan antara aktor dengan kelas *stereotype* ataupun objek lainya |
| 3 |  | Merupakan *entity class* dan merupakan representasi dari sebuah entitas |
| 4 |  | Merupakan *control class* dan merupakan representasi kontrol |
| 5 | :nama objek | Merupakan representasi objek |

4. *Class diagram* adalah jenis statis diagram struktur yang mana di penilitian ini menggambarkan struktur dari suatu sistem dengan menunjukkan sistem kelas, atribut mereka, operasi(atau metode), dan hubungan antara objek-objek (Sparks, 2011).

**Tabel 2.4 Simbol-simbol di dalam *Class Diagram***

| No | Nama symbol | Definisi |
|---|---|---|
| 1 | Kelas<br>nama_kelas<br>+atribut()<br>+operasi() | Merupakan kelas pada struktur sistem |

**Tabel 2.4 Simbol-simbol di dalam Class Diagram (lanjutan)**

| | | |
|---|---|---|
| 2 | *Interface*  | Merupakan konsep *interface* dalam pemrograman beorientasi objek |
| 3 | Asosiasi  | Merupakan relasi antar kelas |
| 4 | Asosiasi berarah  | Merupakan relasi antar kelas yang bermakna kelas yang satu digunakan oleh kelas yang lain |
| 5 | Generalisasi  | Merupakan Relasi antar kelas dengan prinsip generalisasi |
| 6 | Agregasi  | Merupakan relasi antar kelas bermakna semua bagian |
| 7 | Kebergantungan  | Merupakan relasi kelas yang bergantung dengan kelas lain |

## 2.8 Teori Pengujian

Diperlukannya pengujian sistem yang telah dirancang dan diimplementasikan agar mengetahui kesalahan dan segala jenis kemungkinan yang dapat menimbulkan kesalahan sesuai dengan spesifikasi aplikasi yang telah ditentukan. Berdasarkan standar *IEEE*, pengujian perangkat lunak meliputi pengertian aktivitas yang dilakukan dan mengevaluasi kualitas produk dan untuk mengembangkannya dengan melakkan indentifikasi kelemahan dan permasalahan yang terjadi (Simamarta, 2010).

Strategi untuk pengujian perangkat lunak mengintegrasikan metode desain kasus uji perangkat lunak kedalam serangkaian langkah yang disusun dengan baik dan hasilnya adalah konstruksi perangkat bergerak yang berhasil. Strategi pengujian dapat dilakukan melalui pengujian tingkat rendah yaitu pengujian yang dilakukan pada kode program untuk membuktikan bahwa segmen kode sumber yang kecil telah diimplementasikan dengan tepat. Selain itu terdapat pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang tidak sesuai dengan kebutuhan pengguna (Pressman, 2001).

Pada aplikasi pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan ini dilakukan pengujian dengan menggunakan *validity testing* untuk memvalidasi fungsi – fungsi aplikasi mayor yang tidak sesuai dengan kebutuhan pengguna, serta dilakukan *usability testing* untuk mengukur kepuasan akan kemudahan penggunaan dan kebermanfaatan yang diberikan oleh sistem. Berikut merupakan penjelasan dari *validity testing* dan *usability testing.*

### 2.8.1 *Validity Testing*

*Validity testing* aplikasi pencarian rumah sakit di kota Malang berdasarkan preferensi asuransi kesehatan dicapai melalui sederetan pengujian *blackbox* yang menampilkan kesesuaian sistem dengan persyaratan kebutuhan yang telah didefinisikan pada *use case diagram* dan *use case scenario.* Pengujian menguraikan kelas–kelas pengujian yang akan digunakan untuk mengungkap kesalahan ketika terjadi penyesuaian sistem dengan persyaratan. Pengujian dan skenario uji keduanya didesain untuk memastikan apakah semua persyaratan fungsional terpenuhi pada sistem, semua persyaratan kinerja tercapai, dan dokumentasi dibuat dengan benar (Pressman, 2001).

### *2.8.2 Usability Testing*

Pada penelitian ini, *usability testing* dilakukan untuk mengkaji dan menilai seberapa mudah penggunaan aplikasi yang dibuat. Artinya kata "*usability*" sendiri mengacu pada suatu metode untuk melakukan improvisasi terhadap *ease-of use* selama proses perancangan. *Usability testing* mencakup tiga komponen yaitu (Nielsen, 2012):

1. *Learnability*

Semudah apa pengguna dapat mempelajari penggunaan produk tersebut pada pertama kali penggunaan.

2. *Efficiency*

Secepat apa pengguna dapat melakukan tugasnya.

3. *Satisfaction*

Bagaimana tanggapan pengguna terhadap rancangan produk secara keseluruhan.

**1. Kuisioner USE**

Dalam penelitian ini, acuan kuesioner yang dipakai adalah kuesioner USE. Kuesioner USE merupakan media kuesioner untuk mengukur *usability* dengan memakai tiga parameter dari lima komponen *usability testing* yaitu kegunaan (*usefullness*), kepuasan (*satisfaction*) dan kemudahan penggunaan (*ease of use*) karena sesuai dengan pengembangan dalam penelitian ini yang mana akan banyak pengguna yang akan menggunakan aplikasi ini. *Ease of use* merupakan sebuah parameter yang dibagi menjadi dua faktor yaitu kemudahan dalam penggunaan (*ease of use*) dan kemudahan dalam mempelajari aplikasi (*ease of learning*) (Aelani & Falahah, 2012).

Contoh beberapa pertanyaan dalam kuesioner USE adalah sebagai berikut:

*1. Usefulness*

a. Aplikasi ini dalam pengerjaanya memenuhi ekspektasi saya.
b. Aplikasi ini membuat saya menjadi lebih produktif.
c. Aplikasi ini sangat berguna.

2. *Ease of Use*
   a. Aplikasi ini mudah digunakan.
   b. Aplikasi ini *user-friendly*.
   c. Aplikasi ini fleksibel.
3. *Ease of Learning*
   a. Aplikasi ini dapat dengan mudah dan cepat saya pelajari.
   b. Aplikasi ini mudak diingat dalam penggunaanya.
4. *Satisfaction*
   a. Aplikasi ini menyenangkan untuk digunakan.
   b. Saya merasa saya harus memiliki aplikasi ini.

**2. Skala *Likert***

Skala *Likert* dalam penelitian ini digunakan sebagai acuan penilaian dan analisis hasil dalam melakukan survei untuk keperluan *usability testing*. Metode *Likert* merupakan suatu metode penentuan skala dalam pernyataan sikap yang menggunakan distribusi respons sebagai dasar penentuan nilai skalanya. Nilai dari skala *Likert* tergantung dari suatu kebutuhan. Skala *Likert* menjabarkan variabel yang diukur menjadi indikator variabel dimana indikator tersebut kemudian dijadikan sebagai titik tolak untuk menyusun butir-butir instrumen yang dapat berupa pernyataan atau pernyataan (Risnita, 2014).

Jawaban setiap pertanyaan atau pernyataan yang menggunakan skala *Likert* mempunyai gradiasi dari sangat positif sampai sangat negatif yang dapat berupa kata-kata sebagai contoh:

a. Sangat setuju
b. Setuju
c. Netral
d. Tidak setuju
e. Sangat tidak setuju

Untuk keperluan analisis kuantitatif, maka jawaban tersebut dapat diberi skor yang ditunjukkan pada Tabel 2.5.

**Tabel 2.5 Penilaian Jawaban Kuantitatif**

| Jawaban | Skor |
|---|---|
| Sangat setuju | 5 |
| Setuju | 4 |
| Netral | 3 |
| Tidak Setuju | 2 |
| Sangat Tidak Setuju | 1 |

Perhitungan skala *Likert* diterapkan untuk mendapatkan indeks persentase *usability testing*. Persamaan 1 digunakan untuk menghitung total skor dengan nilaiSTS merupakan jumlah dari jawaban sangat tidak setuju, nilaiTS merupakan jumlah jawaban tidak setuju, nilaiN merupakan jumlah dari jawaban netral, nilaiST merupakan jumlah jawaban setuju dan nilaiSS merupakan jumlah jawaban sangat setuju.

Persamaan 2 digunakan untuk menghitung nilai Y yang didapatkan dari perkalian Skor*Likert*Tertinggi dengan JumlahResponden. Dimana dalam penilaian jawaban kuantitatif pada Tabel 2.5. skor tertinggi adalah pada jawaban sangat setuju yang bernilai lima. Persamaan 3 digunakan untuk menghitung indeks persentase yang didapatkan setelah melakukan perhitungan TotalSkor dari persamaan 1 dan Y dari Persamaan 2.

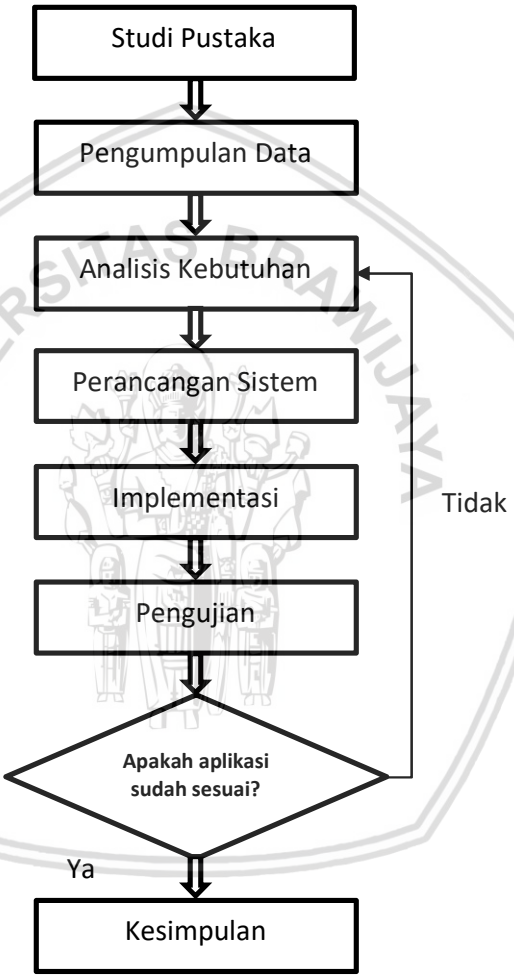
$$TotalSkor = (nilaiSTS \times 1) + (nilaiTS \times 2) + (nilaiN \times 3) + (nilaiST \times 4) + (nilaiSS \times 5) \quad (1)$$

$$Y = SkorLikerTertinggi \times JumlahResponden \quad (2)$$

$$Index(\%) = (TotalSkor/Y) \times 100\% \quad (3)$$

# BAB 3 METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam analisis kebutuhan, perancangan sistem, implementasi dan pengujian dari aplikasi perangkat bergerak yang akan dikembangkan. Kesimpulan dan saran disertakan sebagai catatan atas sistem dan kemungkinan arah pengembangan dimasa yang akan datang. Berikut adalah alur diagram dari metodologi penelitian seperti dalam Gambar 3.1.




**Gambar 3.1 Metodologi Penelitian**

## 3.1 Studi Pustaka

Studi pustaka adalah suatu tahapan dalam penelitian dimana melakukan pengumpulan informasi dan referensi dari sumber pusat informasi seperti *e-book*, jurnal, ataupun *website* resmi. Hasilnya adalah teori-teori pendukung penelitian, seperti berikut ini:

1. Asuransi Kesehatan
2. *Location Based Service*
3. *API Google Maps*
4. Android
5. *Agile System Development*
6. Konsep Rekayasa Aplikasi Perangkat Bergerak
7. Teori Pengujian

## 3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan setelah data yang diperlukan untuk penelitian didapatkan. Kemudian membuat kebutuhan fungsional dan non-fungsional yang dibutuhkan untuk membangun sistem dalam penelitian ini. Kemudian dilakukan identifikasi kebutuhan dilakukan dengan memberi penomoran pada tiap kebutuhan yang telah digali dan digambarkan dengan *Use Case Diagram* dan *Use Case Scenario* agar mengetahui.

### 3.2.1 Pengumpulan Data

Pengumpulan data untuk pembuatan sistem dilakukan dengan survey ke Rumah Sakit di Kota Malang yang dilakukan selama satu bulan dua minggu yang dimulai dari tanggal 15 Maret hingga 13 April 2018. Info yang di ambil dari adalah nama rumah sakit, titik koordinat *latitude* dan *longitude* setiap rumah sakit, nomor telepon rumah sakit dan menanyakan asuransi kesehatan apa saja yang diterima oleh rumah sakit tersebut hanya untuk rawat jalan. *Latitude* dan *longitude* dari rumah sakit dicari menggunakan fitur *Park Here* yang ada di *Google Maps*. Metode ini dirasa cukup baik digunakan di kota besar karena banyaknya menara BTS yang membuat pengambilan *latitude* dan *longitude* dapat akurat. Form survey dengan format dapat ditemukan dalam bagian lampiran (Lampiran A) dan hasil dari survey dapat ditemukan dalam bagian lampiran (Lampiran B).

### 3.2.2 Analisis *Agile System Development*

*Agile System Development* dalam analisis kebutuhan digunakan untuk menentukan kebutuhan fungsional apa saja yang dibutuhkan oleh pengguna. *Agile System Development* sangat baik digunakan karena adanya keterlibatan pengguna secara langsung dalam pembuatannya dan dapat merubah *requirement* sewaktu-waktu yang mana mungkin pengguna membutuhkan fitur tersebut. Iterasi dapat dilakukan berulang-ulang bila pengguna memerlukan suatu kebutuhan fungsional agar mempermudah pengguna dalam pencarian rumah sakit di Kota Malang berdasarkan asuransi kesehatan.

## 3.3 Perancangan Sistem

Tahap perancangan dilakukan untuk mengubah model kebutuhan menjadi model perancangan sistem yang berupa rancangan detail arsitektur perangkat lunak, rancangan struktur data, dan rancangan komponen yang diperlukan untuk mewujudkan sistem. Perancangan menggunakan pendekatan *user experinece* dengan alur proses kerja sistem yang digambarkan dengan perancangan arsitekur sistem, perancangan basis data dan komunikasi, perancangan *uml* yang terdiri dari perancangan *activity diagram*, perancangan *class diagram* dan perancangan *sequence diagram*, serta perancangan aplikasi yang terdiri dari *screenflow, wireframe* dan *mockup UI*.

## 3.4 Implementasi

Implementasi tahap realisasi rancangan perangkat lunak menjadi sistem yang dapat dioperasikan. Tahap ini dilakukan untuk mewujudkan seluruh model yang dihasilkan dalam proses perancangan sebagai kode program, yang awalnya dilakukan spesifikasi sistem dan memberi Batasan implementasi. Pada akhir tahap ini akan didapatkan hasil implementasi basis data, impementasi kode program dan implementasi antarmuka yang dapat dioperasikan oleh pengguna.

### 3.4.1 Implementasi Sistem *Agile System Development*

*Agile System Development* dalam implementasi digunakan untuk menunjukan perubahan dari setiap perubahan fungsional dari setiap iterasi dalam bentuk perancangan antarmuka. Iterasi dapat dilakukan berulang-ulang dengan menambahkan perancangan yang sebelumnya kebutuhan fungsional telah ditambahkan dalam proses analisis kebutuhan.

## 3.5 Pengujian

Pengujian kebutuhan sistem dilakukan dalam dua tingkatan pengujian yaitu *validity testing* dan *usability testing*. Item yang telah dirumuskan dalam analisis kebutuhan akan mencari acuan untuk melakukan *validity testing*. Pengujian yang dilakukan meliputi pengujian dengan metode *blackbox* menggunakan *validity testing* sistem untuk mengecek apakah semua fungsionalitas sudah berjalan seperti yang ada di perancangan. Pengujian kedua dilakukan dengan metode *usability testing* dengan skala *Likert* dengan mengubilan kepada 20 responden yang diperkuat oleh Sauro (2013), yang mana responden memiliki asuransi kesehatan tertentu dan responden mencoba sendiri aplikasi *Hospital Locator* dengan memberikan *task* dalam menjalankan aplikasi yang dilakukan di pemukiman warga. Selanjutnya dari hasil pengujian akan dilakukan analisis untuk mengetahui hasil dari pengujian yang telah dilakukan. Dokumentasi dari *usability testing* dapat ditemukan dalam bagian lampiran (Lampiran C).

## 3.6 Penutup

Proses ini adalah proses penarikan kesimpulan dari hasil keseluruhan proses penelitian serta penarikan saran untuk pembangunan sistem selanjutnya. Penarikan kesimpulan dilakukan untuk memperoleh rangkuman hasil seluruh proses penelitian pengembangan yang dilakukan. Proses penarikan saran dilakukan dengan melakukan evaluasi terhadap seluruh proses yang telah dilalui serta hasil pengembangan yang diperoleh dalam penelitian ini. Pada akhir proses ini akan diperoleh rumusan kesimpulan dan saran penelitian.

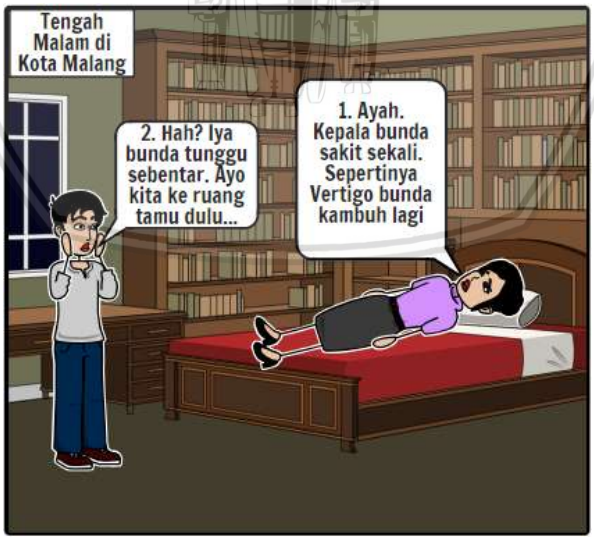
# BAB 4 ANALISIS KEBUTUHAN

Pada tahapan ini dilakukan kegiatan pengumpulan data, gambaran umum aplikasi, fungsi aplikasi, analisis *agile system development,* mengetahui kebutuhan fungsional, pemodelan kebutuhan sistem, dan non-fungsional, yang akan diterapkan sebagai dasar pembangunan sistem. Seluruh tahap tersebut diwujudkan dalam beberapa tahap: Hasil dari tahap ini menjadi jawaban atas pertanyaan penelitian pertama.

## 4.1 Pengumpulan Data

Pengumpulan data dilakukan selama hampir satu bulan dua minggu dengan melakukan survey ke seluruh rumah sakit di Kota Malang dimulai dari tanggal 15 Maret hingga 13 April 2018. Survey dilakukan dengan memberikan surat survey dengan format dapat ditemukan dalam bagian lampiran (Lampiran A). Pencarian data berisikan nama rumah sakit, alamat rumah sakit, nomor telepon rumah sakit, dan asuransi kesehatan yang diterima rumah sakit hanya untuk rawat jalan. *Latitude* dan *longitude* dari rumah sakit dicari menggunakan fitur *Park Here* yang ada di *Google Maps*. Metode ini dirasa cukup baik digunakan di kota besar karena banyaknya menara BTS yang membuat pengambilan *latitude* dan *longitude* dapat akurat. Tujuan dari pengumpulan data adalah sebagai sumber informasi dari aplikasi yang diberi nama *Hospital Locator* yang akan dibuat. dan hasil dari survey dapat ditemukan dalam bagian lampiran (Lampiran B).

## 4.2 Gambaran Umum Aplikasi




**Gambar 4.1 *Story Board Hospital Locator***


**Gambar 4.1 Story Board Hospital Locator (lanjutan)**


**Gambar 4.1 Story Board Hospital Locator (lanjutan)**

Aplikasi yang mana mempunyai fitur utama untuk menampilkan lokasi rumah sakit di Kota Malang. Pengguna dapat memilih untuk melihat lokasi rumah sakit tertentu berdasarkan preferensi asuransi kesehatan atau mellihat lokasi seluruh rumah sakit yang berada di Kota Malang. Selain itu pengguna dapat juga melihat info dari rumah sakit yang ada di Kota Malang yang berisikan nama, alamat, nomor telepon, dan asuransi kesehatan yang diterima rumah sakit yang digambarkan dalam Gambar 4.1 yang menggunakan aplikasi dari *web storyboardthat.com*.

Seperti dalam Gambar 4.1 adalah gambaran sistem dari *aplikasi Hospital Locator* tersedia 24 jam. *Hospital Locator* menampilkan beberapa asuransi kesehatan yang mana sudah diterima di beberapa rumah sakit di Kota Malang. *Hospital Locator* juga dapat merutekan dari lokasi pengguna menuju ke lokasi rumah sakit. Dan *Hospital Locator* juga dapat memiliki inpformasi dari masing-masing rumah sakit.

## 4.3 Fungsi Aplikasi

Secara umum, aplikasi *Hospital Locator* ini memiliki fungsi sebagai berikut:

1. Melakukan pendaftaran diri yang berisikan nama, *email*, *password* & asuransi kesehatan yang dimiliki pengguna sehingga saat pencarian pengguna tidak kesulitan saat mencari asuransi miliknya dan dapat terjaga keamanan profilenya.
2. Menampilkan lokasi rumah sakit di Kota Malang dengan preferensi asuransi kesehatan yang dimiliki oleh pengguna dan lokasi pengguna dengan menggunakan *Maps*
3. Menampilkan lokasi seluruh rumah sakit di Kota Malang dan lokasi pengguna dengan menggunakan *Maps*
4. Menampilkan rute yang dapat dituju dari lokasi pengguna menuju rumah sakit dengan *API Google Maps*.
5. Menampilkan Info dari rumah sakit di Kota Malang yang berisikan nama, alamat, dan nomor telepon rumah sakit, dan asuransi kesehatan yang diterima oleh rumah sakit.

**Tabel 4.1 Tabel Fungsi Produk/Aplikasi**

| No | Identifikasi Pengguna | Karakteristik |
|---|---|---|
| 1 | Pengguna | • Pengguna dapat mendaftarkan dirinya dengan menginputkan nama, *email*, *password* & asuransi kesehatan yang dimiliki pengguna sehingga saat pencarian pengguna tidak kesulitan saat mencari asuransi miliknya dan dapat terjaga keamanan profilenya<br>• Pengguna dapat melihat lokasi rumah sakit dengan berdasarkan preferensi asuransi kesehatan yang dia miliki di Kota Malang dan lokasi pengguna |

**Tabel 4.1 Tabel Fungsi Produk/Aplikasi (lanjutan)**

| No | Identifikasi Pengguna | Karakteristik |
|---|---|---|
| 1 | Pengguna | • Pengguna dapat melihat lokasi seluruh rumah sakit di Kota Malang dan lokasi pengguna<br>• Pengguna dapat melihat rute yang dapat ditempuh dari lokasi pengguna menuju rumah sakit<br>• Pengguna dapat melihat Info dari rumah sakit di Kota Malang yang berisikan nama, alamat, nomor telepon, dan asuransi kesehatan yang diterima rumah sakit |

## 4.4 Analisis *Agile System Development*

Dalam iterasi pertama kebutuhan fungsional aplikasi *Hospital Locator* terdapat delapan fungsional yaitu pencarian rumah sakit berdasarkan asuransi kesehatan, pemilihan asuransi kesehatan, menampilkan lokasi pengguna dan lokasi rumah sakit, melihat detail rumah sakit, merouting dari lokasi pengguna ke rumah sakit, pencarian rumah sakit seluruh Kota Malang, melihat info rumah sakit, pemilihan info rumah sakit.

Setelah pengujian pertama kepada calon pengguna dan masuk dalam tahap iterasi kedua, ada pembaruan dalam fungsional sistem dengan menambahkan lima fitur tambahan yaitu login, register, view, edit dan menelpon rumah sakit yang mana lebih mempermudah pengguna dalam pemakaian aplikasi *Hospital Locator*.

## 4.5 Kebutuhan Fungsional

Kebutuhan fungsional dalam pengembangan ini dirumuskan berdasarkan spesifikasi sistem pencarian lokasi rumah sakit di Kota Malang. Berdasarkan dari hasil kesimpulan dari fungsi produk/aplikasi yang menjadi dasar pembuatan kebutuhan fungsional.

Definisi tersebut dicatat sebagai bagian dari definisi kebutuhan sistem dan diberi kode *FRS*. Daftar definisi kebutuhan fungsional akhir yang diterapkan dalam pembangunan sistem dicantumkan dalam Tabel 4.2.

**Tabel 4.2 Daftar Definisi Kebutuhan Fungsional Sistem**

| Nomor | Definisi Kebutuhan | Use Case | Aktor |
|---|---|---|---|
| FRS_100 | Sistem menyediakan fitur bagi pengguna untuk melakukan autentifikasi *Login* dengan menginputkan *email* dan *password* | *Login* | Pengguna |
| FRS_101 | Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | *Register* | Pengguna |
| FRS_102 | Sistem menyediakan fitur bagi pengguna untuk melihat data diri yang berisikan nama, *email, password*, dan asuransi kesehatan yang dimiliki | *View* | Pengguna |
| FRS_103 | Sistem menyediakan fitur bagi pengguna untuk mengedit data diri yang berisikan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | *Edit* | Pengguna |

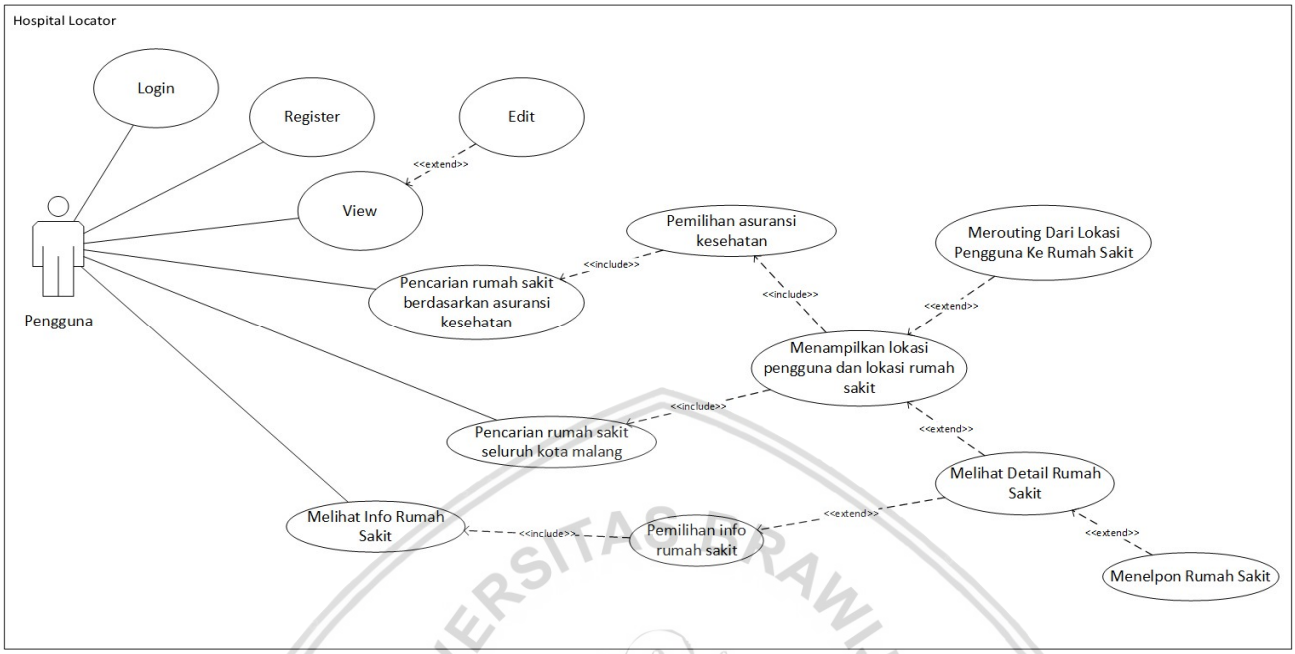
**Tabel 4.2 Daftar Definisi Kebutuhan Fungsional Sistem (lanjutan)**

| | | | |
|---|---|---|---|
| FRS_104 | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit di Kota Malang berdasarkan prefensi asuransi kesehatan yang dimiliki oleh pengguna | Pencarian rumah sakit berdasarkan asuransi kesehatan | Pengguna |
| FRS_105 | Sistem menyediakan fitur bagi pengguna untuk menampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang | Pemilihan asuransi kesehatan | Pengguna |
| FRS_106 | Sistem menyediakan fitur bagi pengguna untuk menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima asuransi kesehatan dalam bentuk *Maps* | Menampilkan lokasi pengguna dan lokasi rumah sakit | Pengguna |
| FRS_107 | Sistem menyediakan fitur bagi pengguna untuk menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* | Melihat detail rumah sakit | Pengguna |

**Tabel 4.2 Daftar Definisi Kebutuhan Fungsional Sistem (lanjutan)**

| | | | |
|---|---|---|---|
| FRS_108 | Sistem menyediakan fitur bagi pengguna untuk dapat menelpon rumah sakit | Menelpon rumah sakit | Pengguna |
| FRS_109 | Sistem menyediakan fitur bagi pengguna untuk menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* | Merouting dari lokasi pengguna ke rumah sakit | Pengguna |
| FRS_110 | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit seluruh Kota Malang bagi pengguna yang tidak memiliki asuransi kesehatan | Pencarian rumah sakit seluruh Kota Malang | Pengguna |
| FRS_111 | Sistem menyediakan fitur bagi pengguna untuk mengetahui info dari masing-masing rumah sakit di Kota Malang | Melihat info rumah sakit | Pengguna |
| FRS_112 | Sistem menyediakan fitur bagi pengguna untuk menampilkan daftar nama rumah sakit seluruh Kota Malang | Pemilihan info rumah sakit | Pengguna |

## 4.6 Pemodelan Kebutuhan




**Gambar 4.2 *Use Case Diagram Sistem***

Berdasarkan penggolongan yang telah di lakukan dalam kebutuhan fungsional, dilakukan proses pemetaan terhadap kebutuhan fungsional sistem. Melalui proses tersebut didapatkan 13 use case fungsi. Pemetaan tersebut, kemudian direpresentasikan sebagai *use case diagram* dalam Gambar 4.2.

Selain direpresentasikan menggunakan *use case diagram*, dalam penelitian ini juga dilakukan spesifikasi *use case* yang dipaparkan menggunakan tabel berisi penjelasan mengenai *aktors*, *objective, pre-condition, main flow, alternative flow* (bila terdapat), *post-condition* dari setiap *use case.* Spesifikasi-spesifikasi *use case* tersebut dipaparkan dalam Tabel 4.3 hingga tabel Tabel 4.14.

**Tabel 4.3 *Use Case Scenario "Login"***

| ***Login*** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan *login* kedalam sistem |
| ***Pre-Condition*** | Aktor berada pada halaman *login* sistem |
| ***Main Flow*** | 1. Aktor dapat memulai *login* dengan menginputkan *email* dan *password* yang mana aktor telah mendaftarkan sebelumnya<br>2. Aktor menekan tombol masuk untuk menjalankan aplikasi *Hospital Locator* |

**Tabel 4.3 Use Case Scenario "Login" (lanjutan)**

| | |
|---|---|
| ***Alternative Flow*** | 1. Bila aktor belum memiliki akun maka aktor harus melakukan registrasi terlebih dahulu<br>2. Bila aktor menginputkan *email*/*password* yang salah maka tidak akan masuk ke dalam aplikasi *Hospital Locator* dan akan muncul pemberitahuan |
| ***Post-Condition*** | Masuk ke dalam aplikasi *Hospital Locator* |

**Tabel 4.4 *Use Case Scenario* "*Register*"**

| | |
|---|---|
| ***Register*** | |
| ***Actors*** | Pengguna |
| ***Objective*** | Aktor yang belum mempunyai akun terlebih dahulu harus melakukan *register* |
| ***Pre-Condition*** | Aktor berada pada halaman *login* sistem |
| ***Main Flow*** | 1. Aktor menekan tombol daftar disini<br>2. Aktor menginputkan nama, *email*, *password* dan asuransi kesehatan yang dimiliki oleh aktor<br>3. Aktor menekan tombol daftar dan akan masuk kembali ke halaman *login* system<br>4. Aktor mengisikan *email* dan *password* untuk masuk ke dalam aplikasi *Hospital Locator* |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Agar dapat melakukan *login* ke dalam aplikasi *Hospital Locator* |

**Tabel 4.5 *Use Case Scenario "View"***

| | |
|---|---|
| ***View*** | |
| ***Actors*** | Pengguna |
| ***Objective*** | Aktor dapat melihat data diri yang sebelumnya aktor telah *register* datanya |
| ***Pre-Condition*** | Aktor berada pada halaman menu utama sistem |
| ***Main Flow*** | 1. Aktor menekan tombol lihat profil<br>2. Aktor diperlihatkan nama, *email*, *password* dan asuransi kesehatan yang dimiliki oleh aktor |

**Tabel 4.5 Use Case Scenario "View" (lanjutan)**

| | |
|---|---|
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Agar dapat melihat data diri yang berisikan nama, *email, password,* dan asuransi kesehatan yang dimiliki aktor |

**Tabel 4.6 *Use Case Scenario* "*Edit*"**

| ***Edit*** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan *edit* di data diri dari pengguna yang sebelumnya aktor telah *register* datanya |
| ***Pre-Condition*** | Aktor berada pada halaman lihat profil |
| ***Main Flow*** | 1. Aktor menekan tombol sunting profil<br>2. Aktor diperlihatkan nama, *email*, *password* dan asuransi kesehatan yang dimiliki oleh aktor<br>3. Aktor dapat merubah salah satu data dan memasukan inputan baru<br>4. Aktor menekan tombol simpan profil |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Aktor dapat melakukan perubahan data yang sebelumnya aktor telah *register* datanya |

**Tabel 4.7 *Use Case Scenario* "Pemilihan Asuransi Kesehatan"**

| **Pemilihan asuransi kesehatan** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan pencarian nama asuransi kesehatan yang dimiliki oleh aktor |
| ***Pre-Condition*** | Aktor berada pada halaman pencarian rumah sakit berdasarkan asuransi kesehatan |
| ***Main Flow*** | 1. Aktor akan ditampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang dan akan memberi tanda di salah satu asuransi yang telah dipilih oleh aktor dalam daftar agar tidak kesulitan mencari asuransinya. |

**Tabel 4.7 Use Case Scenario "Pemilihan Asuransi Kesehatan" (lanjutan)**

| | |
|---|---|
| | 2. Aktor dapat memulai pencarian asuransi kesehatan yang dimiliki oleh aktor dengan memilih dari daftar asuransi kesehatan yang telah disediakan oleh sistem<br>3. Aktor memilih asuransi kesehatan yang dimilikinya |
| ***Alternative Flow*** | 1. Aktor dapat mencari manual nama asuransi kesehatan dengan menginputkan nama asuransi kesehatan yang dimiliki oleh aktor di *search bar* dan memilihnya<br>2. Aktor dapat memilih lebih dari satu asuransi untuk melihat rumah sakit yang menerima asuransi kesehatan |
| ***Post-Condition*** | Menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima asuransi kesehatan dalam bentuk *Maps* |

**Tabel 4.8 *Use Case Scenario* "Menampilkan Lokasi Pengguna Dan Lokasi Rumah Sakit Berdasarkan Asuransi Kesehatan"**

| **Menampilkan lokasi pengguna dan lokasi rumah sakit berdasarkan asuransi kesehatan** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Mengetahui rumah sakit mana saja yang menerima asuransi kesehatan yang telah dipilih oleh aktor sebelumnya |
| ***Pre-Condition*** | Aktor berada pada halaman *maps* pencarian rumah sakit / halaman awal sistem |
| ***Main Flow*** | 1. Aktor akan ditampilkan lokasi dia berada saat itu dan lokasi beberapa lokasi rumah sakit<br>2. Aktor dapat memulai pencarian rumah sakit telah disediakan oleh sistem<br>3. Aktor memilih rumah sakit |
| ***Alternative Flow*** | 1. Dapat melihat detail dari rumah sakit dengan menekan deskripsi nama dan alamat rumah sakit<br>2. Dapat merouting dari lokasi pengguna ke lokasi rumah sakit menggunakan *API Google Maps* |
| ***Post-Condition*** | Menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* |

**Tabel 4.9 *Use Case Scenario* "Melihat Detail Rumah Sakit"**

| Melihat detail rumah sakit | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Mengetahui nama, alamat, nomor telepon dan asuransi apa saja yang diterima di rumah sakit |
| ***Pre-Condition*** | Aktor berada pada halaman *maps* pencarian rumah sakit |
| ***Main Flow*** | 1. Aktor akan ditampilkan nama, alamat, nomor telepon dan asuransi apa saja yang diterima di rumah sakit yang telah dipilih oleh aktor sebelumnya setelah menekan deskripsi nama dan alamat rumah sakit<br>2. Aktor menekan tombol *routing* untuk menunjukan rute dari lokasi aktor menuju ke lokasi rumah sakit |
| ***Alternative Flow*** | Menelpon rumah sakit |
| ***Post-Condition*** | Menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* |

**Tabel 4.10 *Use Case Scenario* "Menelpon Rumah Sakit"**

| Melihat detail rumah sakit | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Menelpon rumah sakit yang telah dipilih oleh aktor sebelumnya |
| ***Pre-Condition*** | Aktor berada pada halaman detail rumah sakit |
| ***Main Flow*** | 1. Aktor akan dapat menelpon rumah sakit bila menekan nomor telpon yang tersedia di halaman detail rumah sakit<br>2. Aktor di tampilkan halaman *calling* yang mana sudah tersedia nomor telepon rumah sakit<br>3. Aktor menelpon rumah sakit |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Aktor menelpon rumah sakit yang telah dipilih oleh aktor sebelumnya |

**Tabel 4.11 *Use Case Scenario* "*Routing* Rumah Sakit Berdasarkan Asuransi Kesehatan"**

| ***Routing* rumah sakit berdasarkan asuransi kesehatan** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Menuju ke lokasi rumah sakit dari lokasi aktor berada |
| ***Pre-Condition*** | Aktor berada pada halaman *maps* pencarian rumah sakit / detail rumah sakit |
| ***Main Flow*** | 1. Aktor menekan alamat yang tersedia di halaman detail rumah sakit<br>2. Aktor akan ditampilkan halaman *API Google Maps* yang telah tersedia *routing* dari lokasi aktor menuju lokasi rumah sakit yang telah dipilih aktor sebelumnya<br>3. Aktor akan menekan tombol *start* untuk memulai *routing* dari lokasi aktor menuju lokasi rumah sakit |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Sampai pada rumah sakit yang telah dipilih oleh aktor sebelumnya |

**Tabel 4.12 *Use Case Scenario* "Pencarian Rumah Sakit Seluruh Kota Malang"**

| **Pencarian rumah sakit seluruh Kota Malang** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan pencarian rumah sakit seluruh Kota Malang |
| ***Pre-Condition*** | Aktor berada pada halaman awal dari sistem |
| ***Main Flow*** | Aktor dapat memulai pencarian rumah sakit seluruh Kota Malang setelah menekan menu pencarian rumah sakit seluruh Kota Malang |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Menampilkan lokasi pengguna dan lokasi rumah sakit seluruh Kota Malang dalam bentuk *Maps* |

**Tabel 4.13 *Use Case Scenario* "Melihat Info Rumah Sakit"**

| **Melihat Info Rumah Sakit** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan pencarian info rumah sakit |

**Tabel 4.13 Use Case Scenario "Melihat Info Rumah Sakit" (lanjutan)**

| | |
|---|---|
| ***Pre-Condition*** | Aktor berada pada halaman awal dari sistem |
| ***Main Flow*** | Aktor dapat memulai pencarian info rumah sakit setelah menekan menu info rumah sakit |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Menampilkan daftar nama rumah sakit seluruh Kota Malang |

**Tabel 4.14 *Use Case Scenario* "Pemilihan Info Rumah sakit"**

| **Pemilihan info rumah sakit** | |
|---|---|
| ***Actors*** | Pengguna |
| ***Objective*** | Melakukan pencarian nama rumah sakit yang ingin diketahui infonya |
| ***Pre-Condition*** | Aktor berada pada halaman info rumah sakit |
| ***Main Flow*** | 1. Aktor akan ditampilkan daftar seluruh nama rumah sakit seluruh Kota Malang<br>2. Aktor dapat memulai pencarian rumah sakit yang ingin diketahui infonya dengan memilih dari daftar rumah sakit yang telah disediakan oleh sistem<br>3. Aktor memilih rumah sakit yang ingin diketahui infonya |
| ***Alternative Flow*** | Aktor dapat mencari manual nama rumah sakit dengan menginputkan nama rumah sakit yang ingin diketahui infonya di *search bar* dan memilihnya |
| ***Post-Condition*** | Menampilkan info gambar, nama, alamat, nomor telepon, dan daftar asuransi kesehatan yang diterima rumah sakit |

## 4.7 Kebutuhan Non-Fungsional

Analisis kebutuhan non fungsional adalah analisis untuk mengetahui spesifikasi yang dibutuhkan oleh sistem. Sistem pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan ini lebih menekankan pada bagaimana rancangan dan implementasi aplikasi pencarian dan pemberian informasi dapat memberikan kemudahan dan manfaat penggunaan dari sistem terhadap pencarian informasi saat terjadi permasalahan kesehatan. Pengujian yang digunakan untuk mengukur kepuasaan tersebut ialah *usability testing* yang diharapkan dari hasil pengujian tersebut dapat mendapatkan *feedback* dari pengguna yang dapat di iterasi dalam penelitian ini.

# BAB 5 PERANCANGAN SISTEM

Pada bab ini akan membahas mengenai perancangan aplikasi yang akan di bangun. Perancangan yang akan membahas lebih detail tentang perancangan aplikasi berdasarkan analisis kebutuhan yang telah dilakukan sebelumnya. Seluruh tahap tersebut diwujudkan dalam beberapa tahap: Hasil dari tahap ini menjadi jawaban atas pertanyaan penelitian pertama.

## 5.1 Perancangan Arsitektur Sistem

Sistem pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan dibangun dengan menggunakan konsep *client* dan *server*. Perancangan arsitektur sistem dibuat untuk menjelaskan bagaimana arsitektur sistem akan diimplementasikan. Dalam perancangan arsitektur sistem ini menggunakan pendekatan *Location Based Service* (LBS) seperti yang digambarkan pada Gambar 5.1.

Sistem *client* dibuat dengan bahasa pemrograman Java dan XML yang merupakan bagian dari pembuatan aplikasi *native* dari Android dengan memanfaatkan beberapa sensor dari *smartphone* Android tersebut. Sensor yang akan digunakan antara lain *GPS*, dan internet yang berguna untuk client. Sedangkan untuk sistem *server* dibangun dengan menggunakan bahasa pemrograman PHP, dengan menggunakan *Framework Codeigniter* karena menggunakan *pattern* MVC yang mana sesuai dengan perancangan dalam penelitian ini. *Web Service* digunakan untuk menjembatani komunikasi antara *client* dan *server*, dimana proses pertukaran data yang terjadi antara *client* dan *server* dikirim menggunakan format JSON. Admin ditugaskan untuk mengupload semua data yang dibutuhkan oleh *database* untuk aplikasi *Hospital Locator*.


**Gambar 5.1 Arsitektur Sistem Pencarian Rumah Sakit di Kota Malang Berdasarkan Preferensi Kesehatan**

## 5.2 Perancangan *UML*

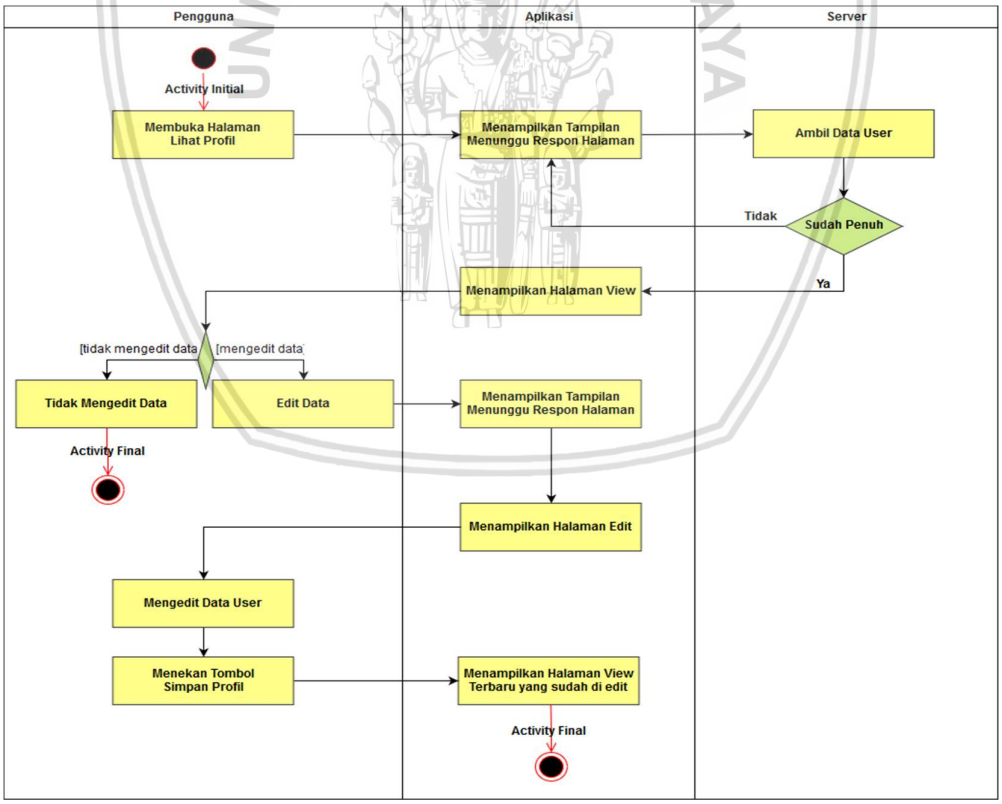
*Unified Modelling Language* (*UML*) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. *UML* sesuai dengan kata terakhir dari kepanjangannya, *UML* itu adalah salah satu bentuk language atau bahasa. Menurut pencetusnya, *UML* di definisikan sebagai bahasa visual untuk menjelaskan, memberikan spesifikasi sistem.

### 5.2.1 Perancangan *Activity Diagram*

*Activity diagram* digunakan untuk memodelkan aktifitas antara pengguna dan sistem yang berjalan berdasarkan pada *use case scenario* yang telah dibuat dalam Tabel 4.3 hingga Tabel 4.14. Berikut ialah *activity* diagram pada sistem pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan.

**1. *Activity Diagram View & Edit***

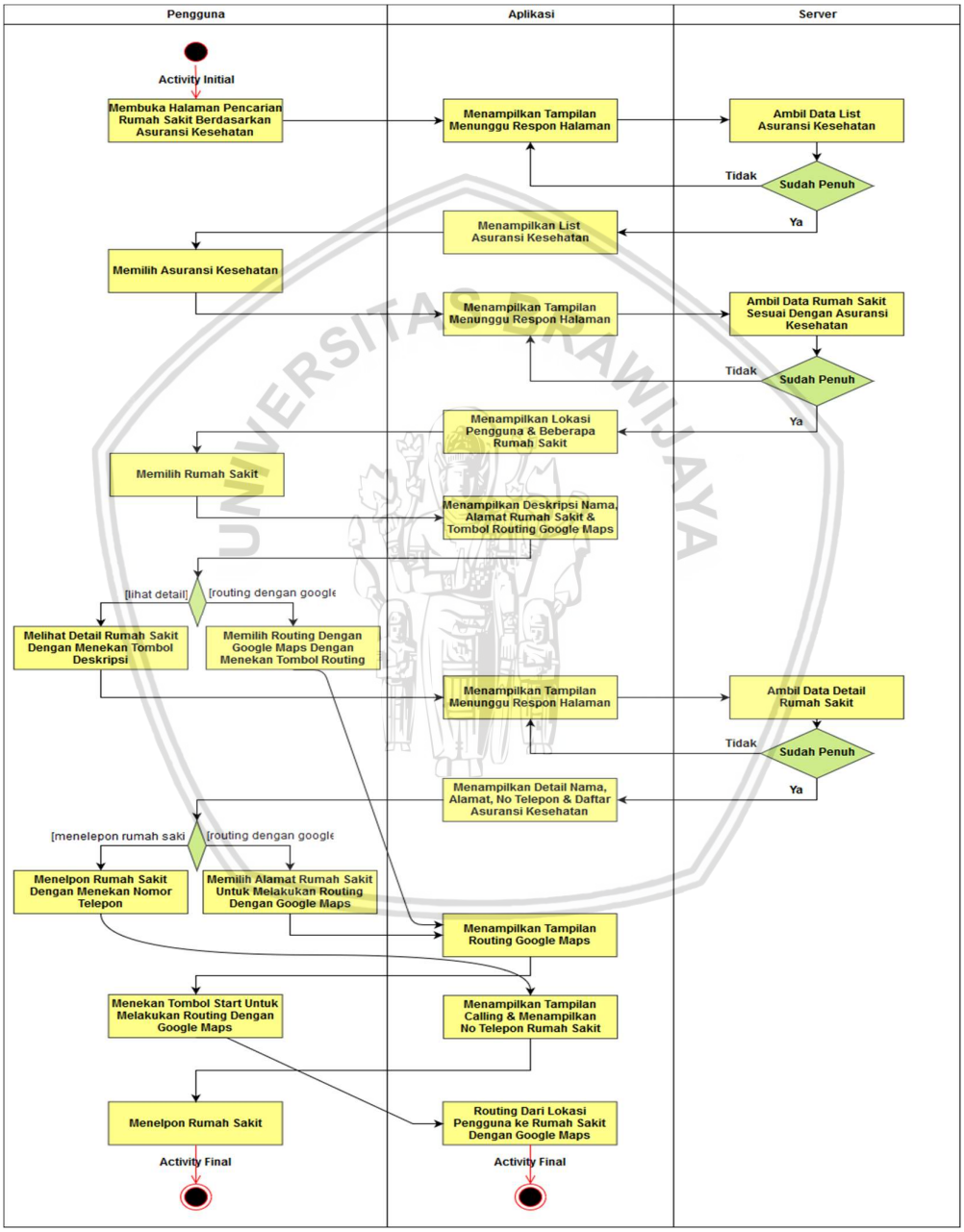
Pada Gambar 5.2 menjelaskan alur dalam mengakses view dan edit data menggunakan aplikasi, pada saat menekan tombol lihat profil didalam menu utama.




**Gambar 5.2 *Activity Diagram View & Edit***

## 2. *Activity Diagram* Pencarian Rumah Sakit Berdasarkan Preferensi Asuransi Kesehatan

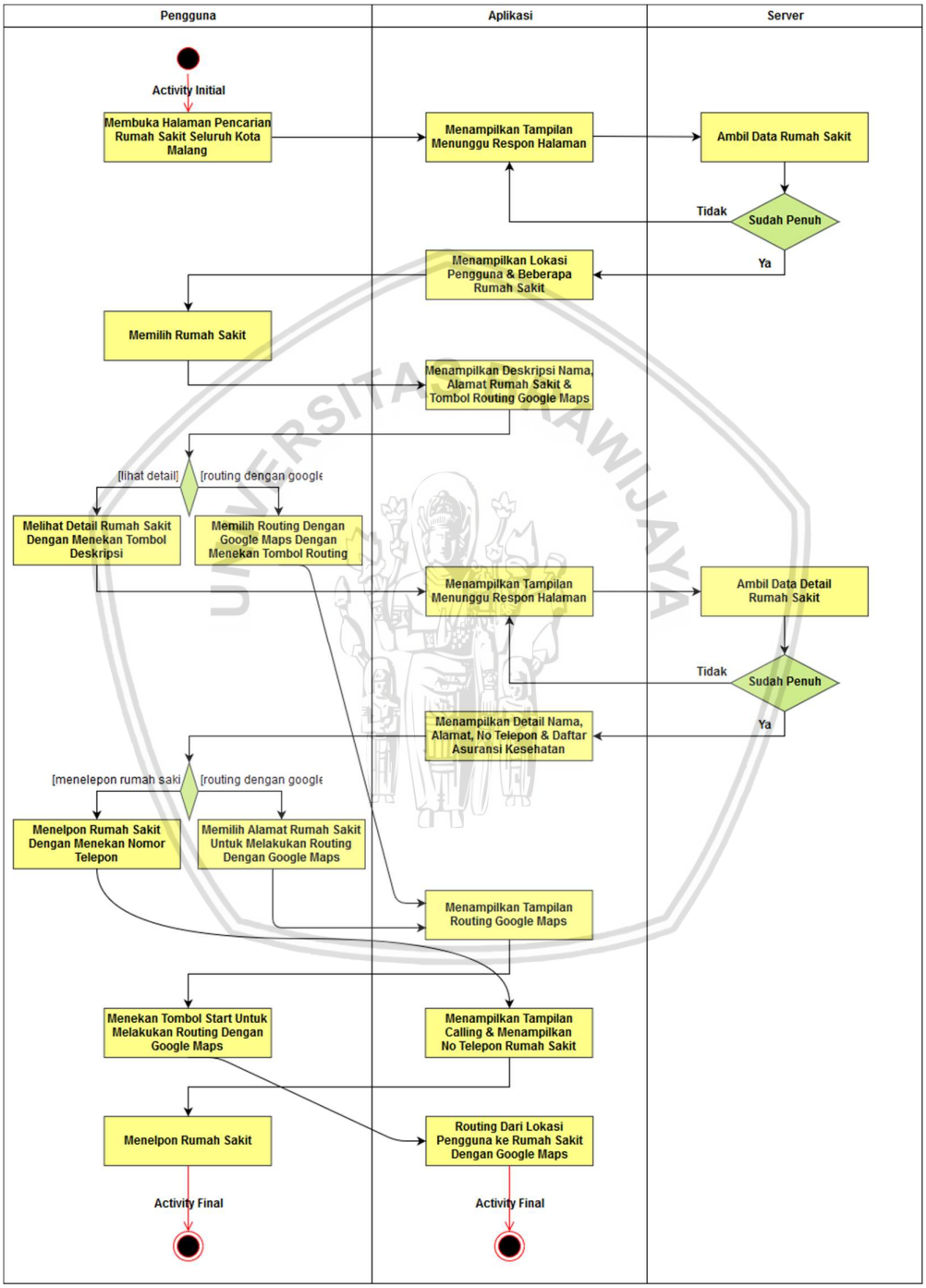
Pada Gambar 5.3 menjelaskan alur dalam mengakses pencarian rumah sakit berdasarkan preferensi asuransi kesehatan menggunakan aplikasi, pada saat menekan tombol pencarian rumah sakit berdasarkan preferensi asuransi kesehatan didalam menu utama.




**Gambar 5.3 *Activity Diagram* Pencarian Rumah Sakit Berdasarkan Preferensi Asuransi Kesehatan**

### 3. *Activity Diagram* Pencarian Rumah Sakit Seluruh Kota Malang

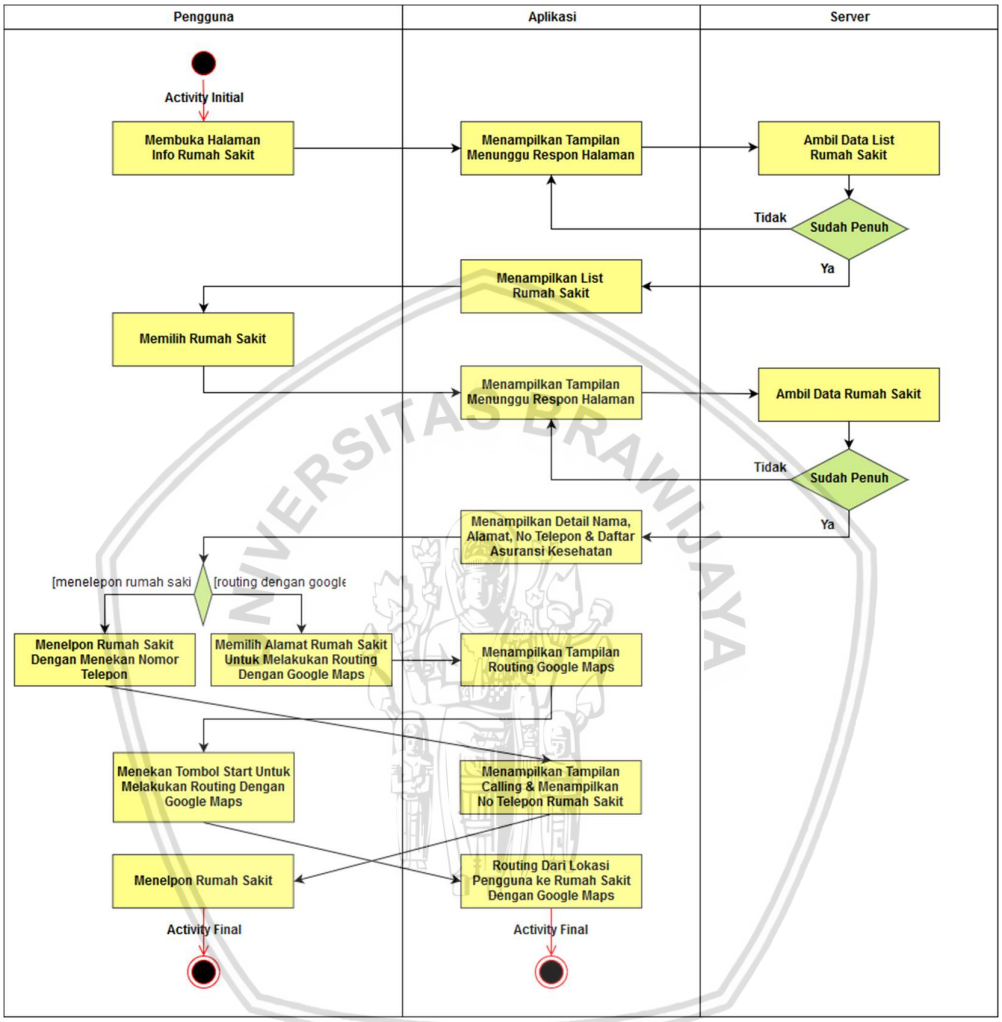
Pada Gambar 5.4 menjelaskan alur dalam mengakses pencarian rumah sakit seluruh Kota Malang menggunakan aplikasi, pada saat menekan tombol pencarian rumah sakit seluruh Kota Malang didalam menu utama.




**Gambar 5.4 *Activity Diagram* Pencarian Rumah Sakit Seluruh Kota Malang**

### 4. *Activity Diagram* Melihat Info Rumah Sakit

Pada Gambar 5.5 menjelaskan alur dalam mengakses melihat info rumah sakit menggunakan aplikasi, pada saat menekan tombol info rumah sakit didalam menu utama.

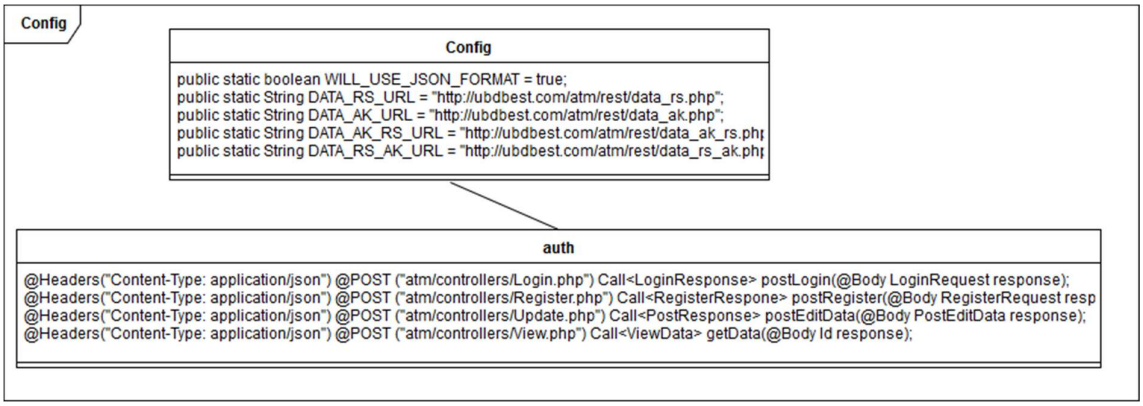



**Gambar 5.5 *Activity Diagram* Info Rumah Sakit**

## 5.2.2 Perancangan *Class Diagram*

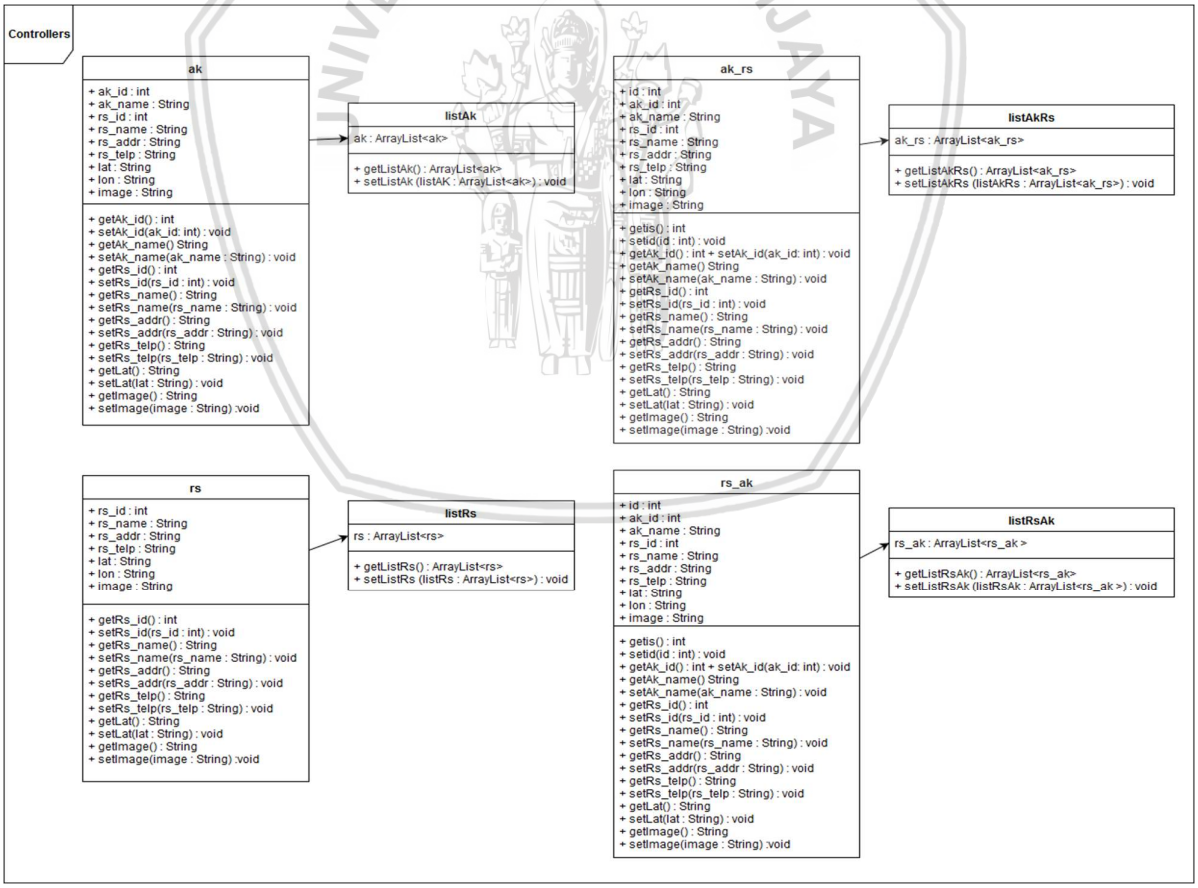
Perancangan *class diagram* digunakan untuk menampilkan sejumlah kelas dan *package* yang ada pada sistem perangkat lunak yang dikembangkan. *Class diagram* merupakan diagram yang menunjukkan hubungan antar kelas dalam sistem yang sedang dibangun dan membentuk suatu relasi. Perancangan ini akan menampilkan perancangan *class diagram* fitur fitur utama dari aplikasi *client*.

Pada Gambar 5.6 ditunjukan *class diagram* pada aplikasi yang merupakan dari *Config*. Dimana terdapat *class Config* dan *auth. Config* dan *auth* adalah penyedia layanan dari masuknya *database* dari *server* yang disimpan sementara di kelas *config* dan *auth*.



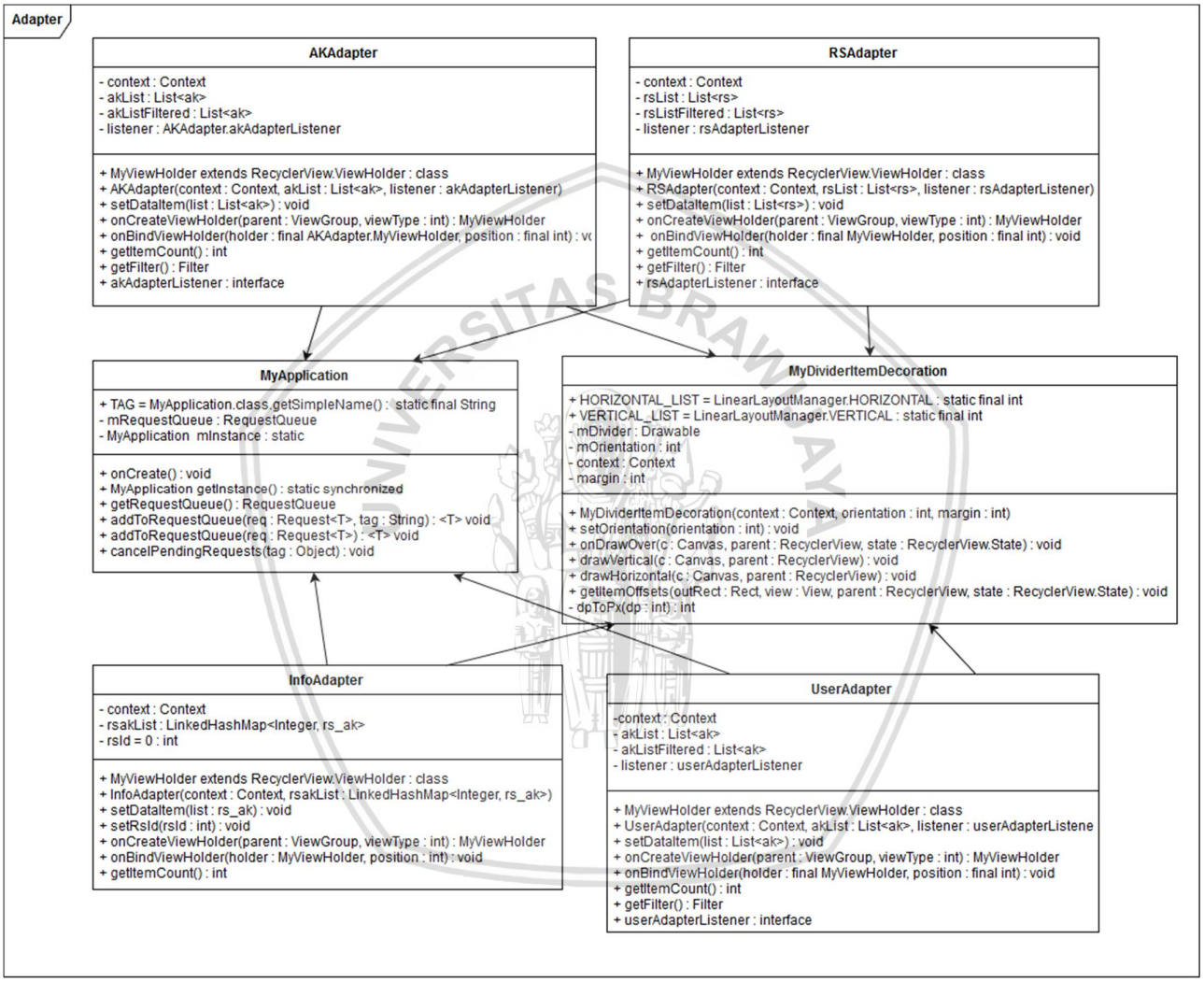

**Gambar 5.6 *Class Diagram Config***

Pada Gambar 5.7 ditunjukan *class diagram* pada aplikasi yang merupakan kumpulan dari *Controllers*. Dimana terdapat *class ak, rs, ak_rs, rs_ak, listAk, listRs, listAkRs,* dan *listRsAk. ak, rs, ak_rs,* dan *rs_ak* adalah tempat inisiasi dari *database* yang sebelumnya disimpan sementara di kelas *config*. *listAk, listRs, listAkRs,* dan *listRsAk* adalah kelas yang menampilkan list dari kelas *ak, rs, ak_rs, rs_ak* yang telah diinisiasi di dalam masing-masing kelas.



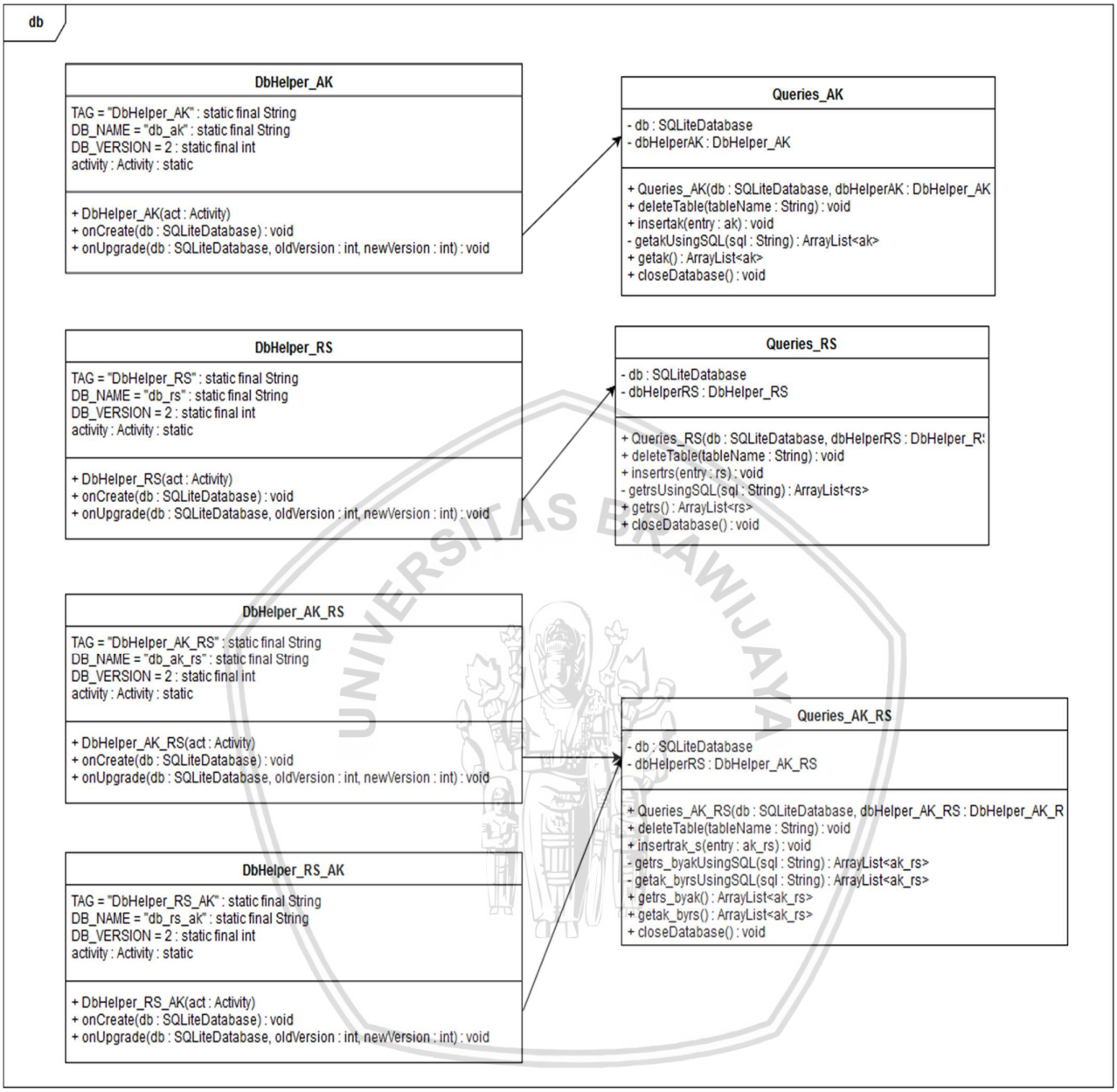

**Gambar 5.7 *Class Diagram Controllers***

Pada Gambar 5.8 ditunjukan *class diagram* pada aplikasi yang merupakan kumpulan dari *Adapter*. Dimana terdapat *AKAdapter, RSAdapter, InfoAdapter, MyApplication,* dan *MyDividerItemDecoration*. *AKAdapter, RSAdapter, UserAdapter* dan *InfoAdapter* adalah kelas yang menyediakan tampilan yang akan di kelas yang berada di *package models*. *MyApplication* dan *MyDividerItemDecoration* adalah penyedia layanan bagi keempat kelas *AKAdapter, RSAdapter, UserAdapter* dan *InfoAdapter*.



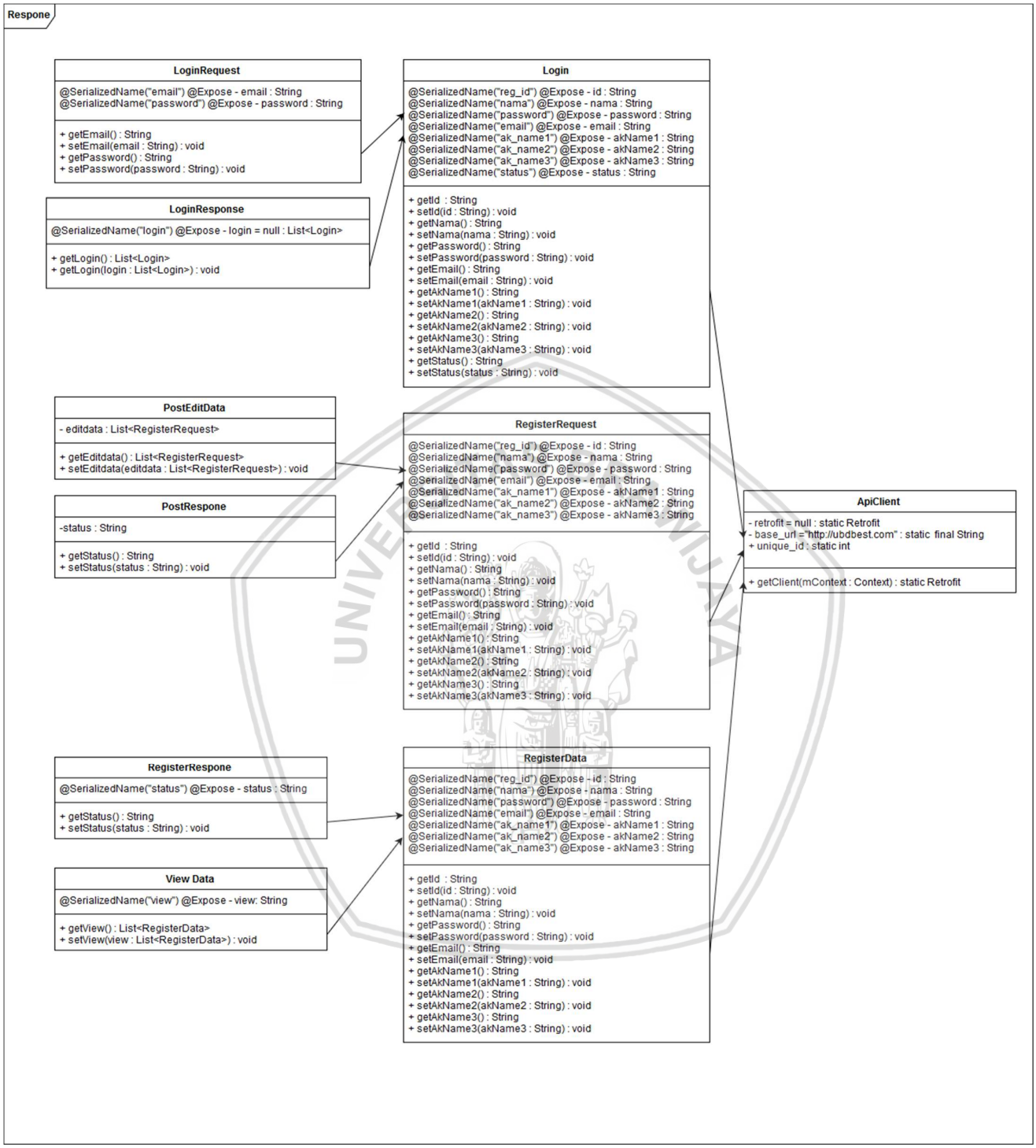

**Gambar 5.8 *Class Diagram Adapter***

Pada Gambar 5.9 ditunjukan *class diagram* pada aplikasi yang merupakan kumpulan dari *db*. Dimana terdapat *class DbHelper_AK, DbHelper_RS, DbHelper_AK_RS, DbHelper_RS_AK, Queris_AK, Queris_RS,* dan *Queris_AK_RS. DbHelper_AK, DbHelper_RS, DbHelper_AK_RS,* dan *DbHelper_RS_AK* adalah pembuat *database* ke dalam Android yang sebelumnya datanya telah diinisiasi di kelas *ak, rs, ak_rs,* dan *rs_ak. DbHelper_AK, DbHelper_RS,* dan *DbHelper_AK_RS* adalah penyedia kode yang mana dapat "*SELECT FROM*" yang dapat dipanggil di *models*.




**Gambar 5.9 *Class Diagram* db**

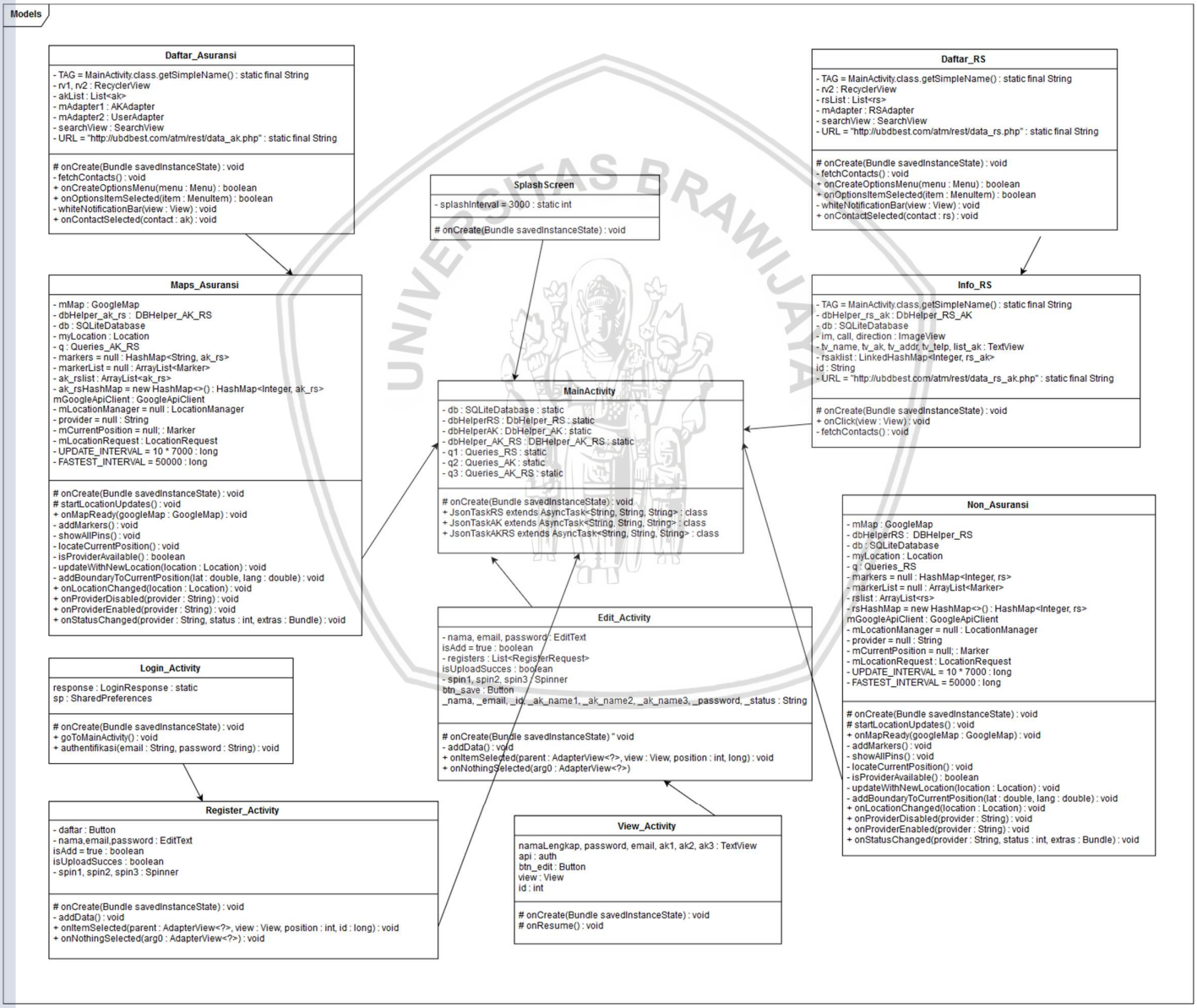
Pada Gambar 5.10 ditunjukan *class diagram* pada aplikasi yang merupakan kumpulan dari *Respone*. Dimana terdapat *class ApiClient, Login, LoginRequest, LoginRespone, PostEditData, PostRespone, RegisterData, RegisterRequest, RegisterRespone, ViewData.* Yang mana digunakan sebagai database sementara dan diinisiasikan dikelas itu untuk dipakai pada kelas *Login_Activity, Register_Activity, View_Activity* & *Edit_Activity.* Masing-masing dari tiap *class* yang ada di *package Respone* merepresentasikan setiap *class* yang ada di kelas *Login_Activity, Register_Activity, View_Activity* & *Edit_Activity.*




**Gambar 5.10 *Class Diagram Respone***

Pada Gambar 5.11 ditunjukan *class diagram* pada aplikasi yang merupakan kumpulan dari *Models*. Dimana terdapat *class Daftar_Asuransi, Daftar_RS, Info_RS, Main_Activity, Maps_Asuransi, Non_Asuransi,* dan *SplashScreen. Daftar_Asuransi, Daftar_RS* dan *Info_RS* digunakan untuk menampilkan tampilan

yang datanya diambil dari *AKAdapter, RSAdapter, UserAdapter* dan *InfoAdapter* yang mana sebagai sourcenya. *Main_Activity* adalah penghubung *database* dari Android ke aplikasi yang mana data yang telah diinsiasi sebelumnya dimasukan ke dalam aplikasi. *Maps_Asuransi* digunakan untuk menampilkan rumah sakit berdasarkan asuransi kesehatan yang mana data tersebut didapatkan dari kelas *DBHelper_AK_RS* dan *Queries_AK_RS*. *Non_Asuransi* digunakan untuk menampilkan rumah sakit seluruh Kota Malang, hampir sama seperti kelas *Maps_Activity* tetapi mengambil data dari kelas *DBHelper_RS* dan *Queries_RS. SplashScreen* digunakan untuk menampilkan tampilan *splash screen* pada awal aplikasi dibuka.



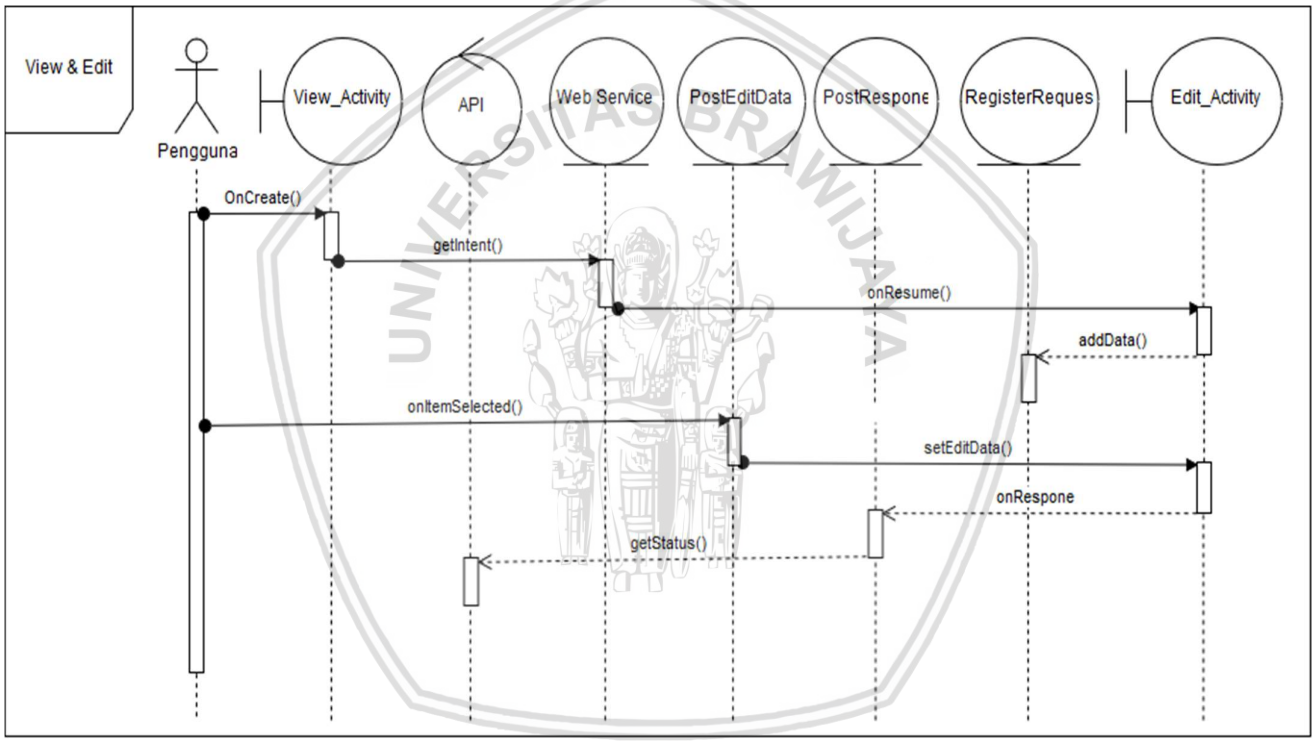

**Gambar 5.11 *Class Diagram Models***

### 5.2.3 Perancangan *Sequence Diagram*

Pada *sequence diagram* akan memodelkan bagaimana perilaku yang dilakukan antara aktor didalam sistem berdasarkan waktu yang runtut. Berikut merupakan diagram-diagram *sequence* pada sistem pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan berdasarkan dari *use case scenario* yang telah dibuat sebelumnya dalam Tabel 4.3 hingga Tabel 4.14.

**1. Sequence Diagram *View & Edit***

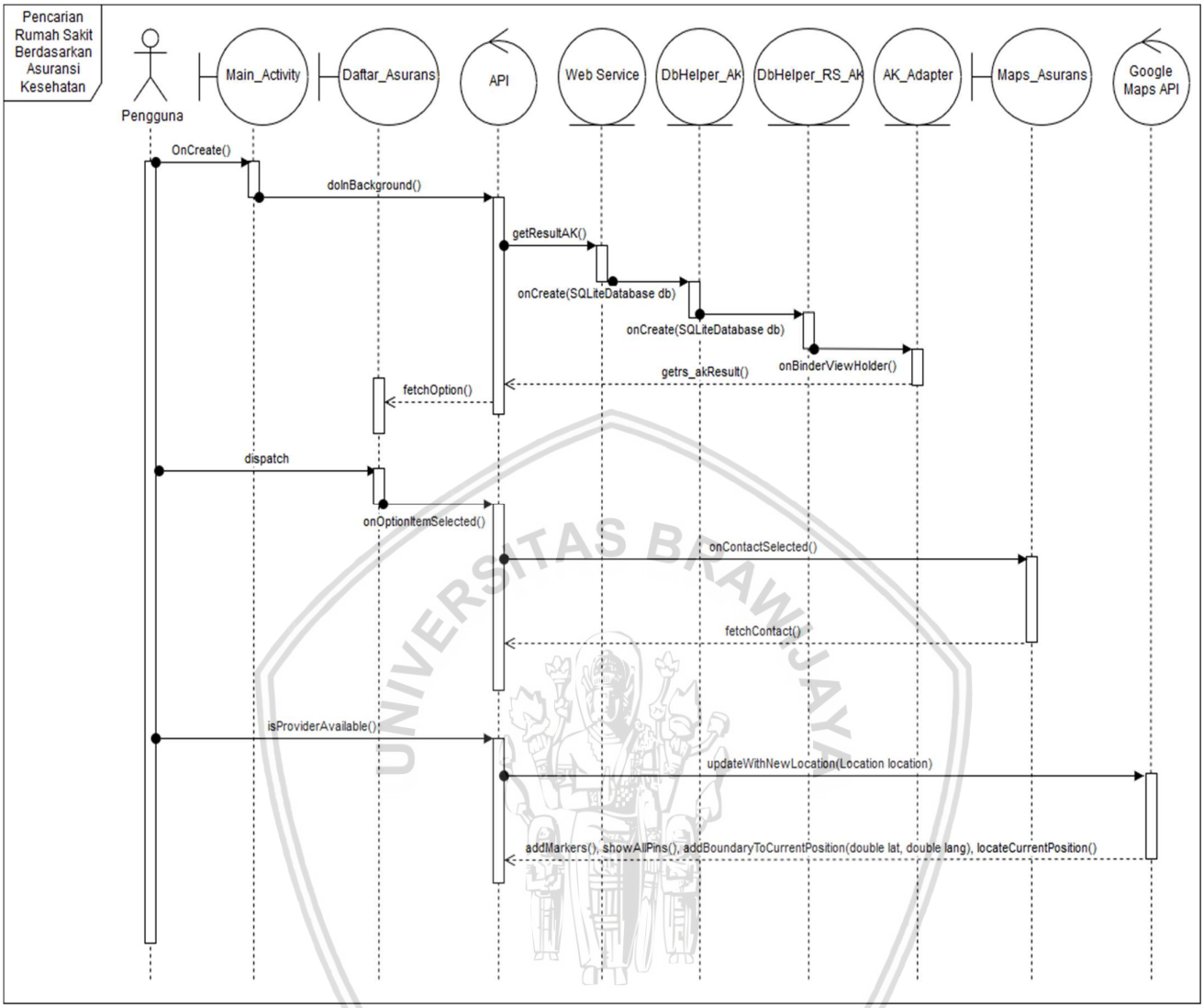
Gambar 5.12 merupakan diagram sequence untuk *login* dan *register*. Diagram ini menggambarkan interaksi ketika pengguna ingin melakukan *login* ke dalam aplikasi dan mendaftar melalui aplikasi dan selanjutnya aplikasi mengambil data dari server untuk masuk ke dalam aplikasi.




**Gambar 5.12 *Sequence Diagram Login* & *Register***

**2. Sequence Diagram Pencarian Rumah Sakit Berdaasarkan Asuransi Kesehatan**

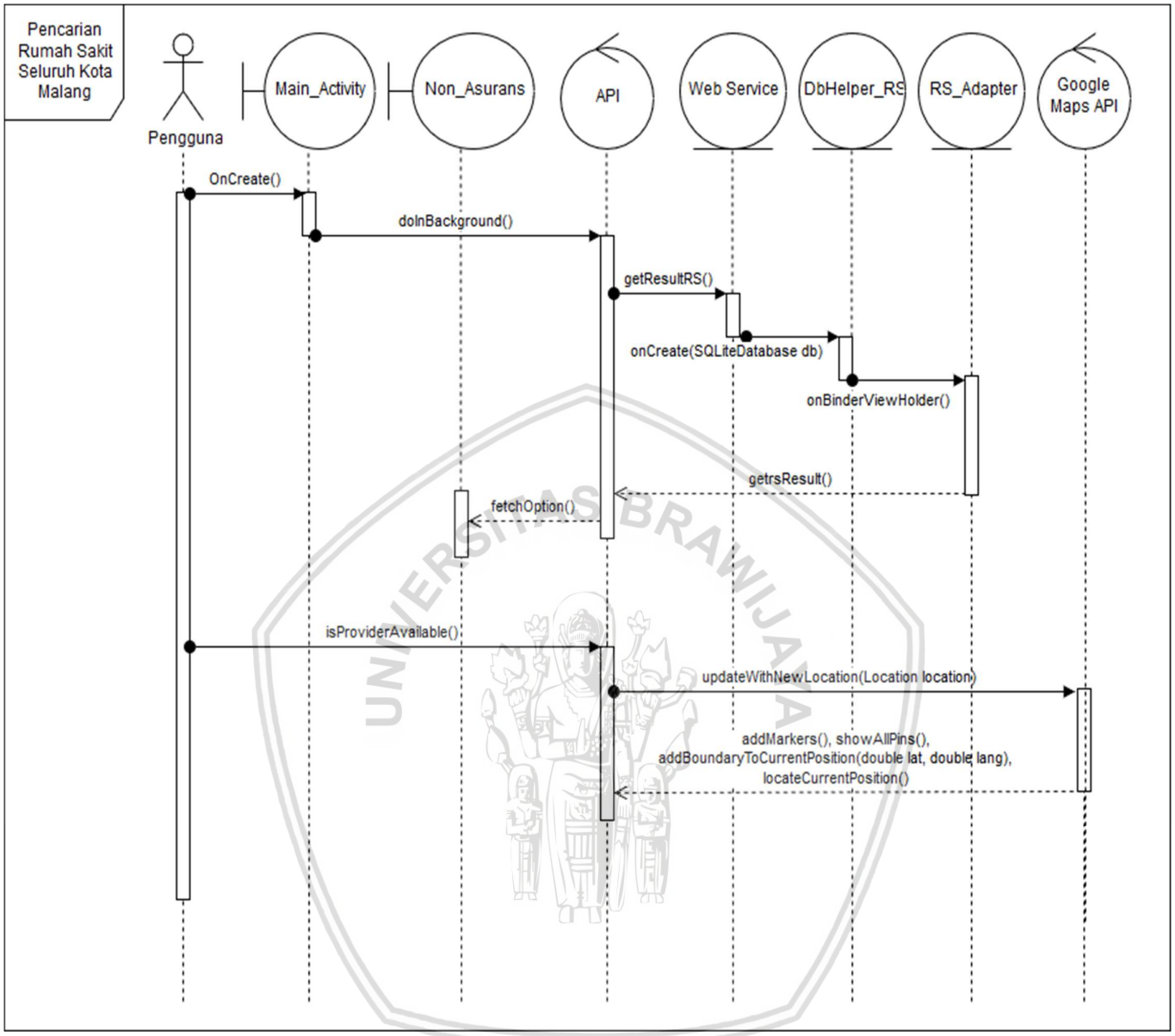
Gambar 5.13 merupakan diagram sequence untuk proses pencarian rumah sakit berdasarkan asuransi kesehatan. Diagram ini menggambarkan interaksi ketika pengguna ingin mencari rumah sakit dengan preferensi kesehatan yang dimiliki oleh pengguna melalui aplikasi dan selanjutnya aplikasi mengambil data dari server untuk ditampilkan.




**Gambar 5.13 *Sequence Diagram* Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan**

### 3. Sequence Diagram Pencarian Rumah Sakit Seluruh Kota Malang

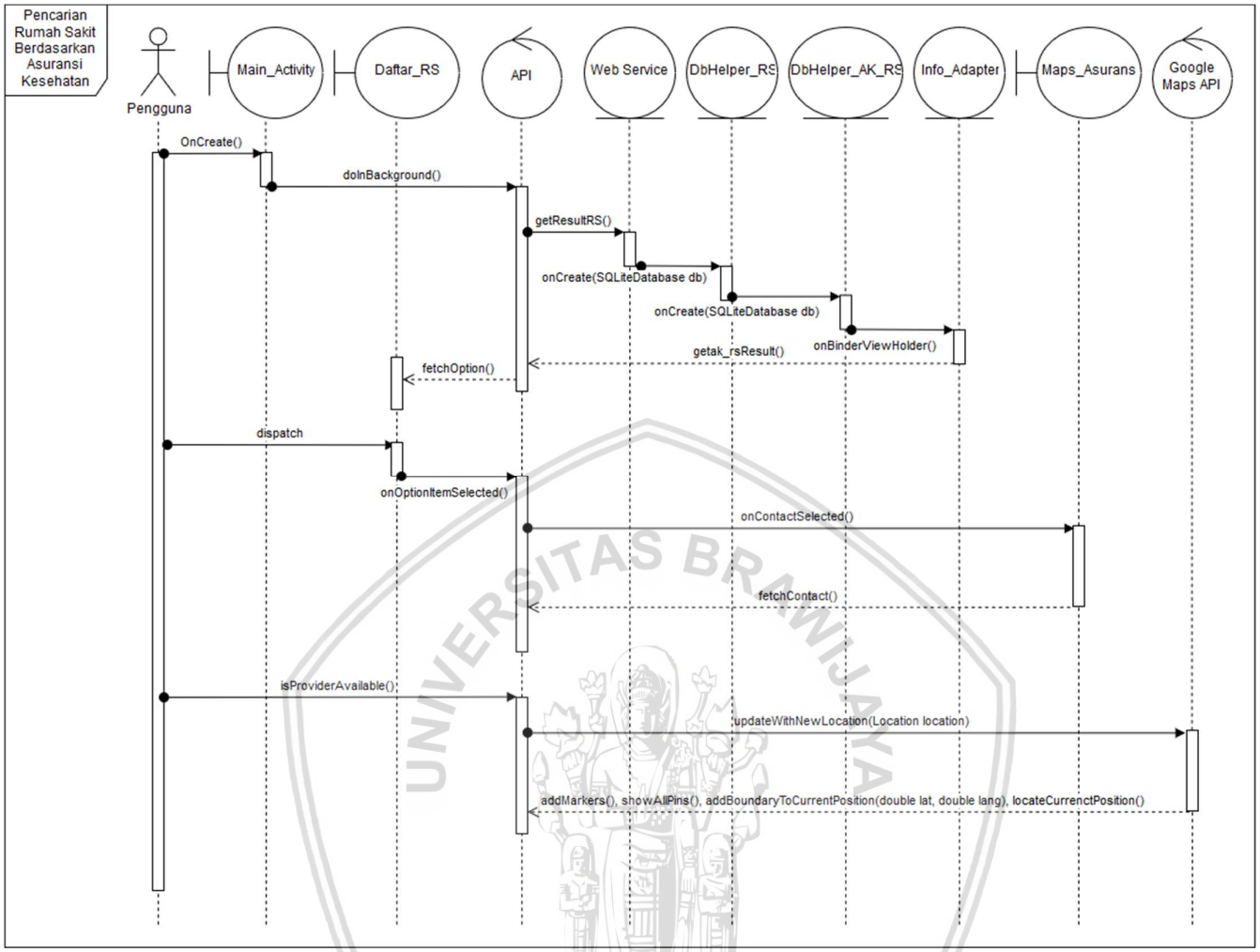
Gambar 5.14 merupakan diagram sequence untuk proses pencarian rumah sakit seluruh Kota Malang. Diagram ini menggambarkan interaksi ketika pengguna ingin mencari rumah sakit yang berada di Kota Malang melalui aplikasi dan selanjutnya aplikasi mengambil data dari server untuk ditampilkan.




**Gambar 5.14 *Sequence Diagram* Pencarian Rumah Sakit Seluruh Kota Malang**

### 4. Sequence Diagram Info Rumah Sakit

Gambar 5.15 merupakan diagram sequence untuk proses melihat info rumah sakit. Diagram ini menggambarkan interaksi ketika pengguna ingin mengetahui info rumah sakit yang berada di Kota Malang melalui aplikasi dan selanjutnya aplikasi mengambil data dari server untuk ditampilkan.



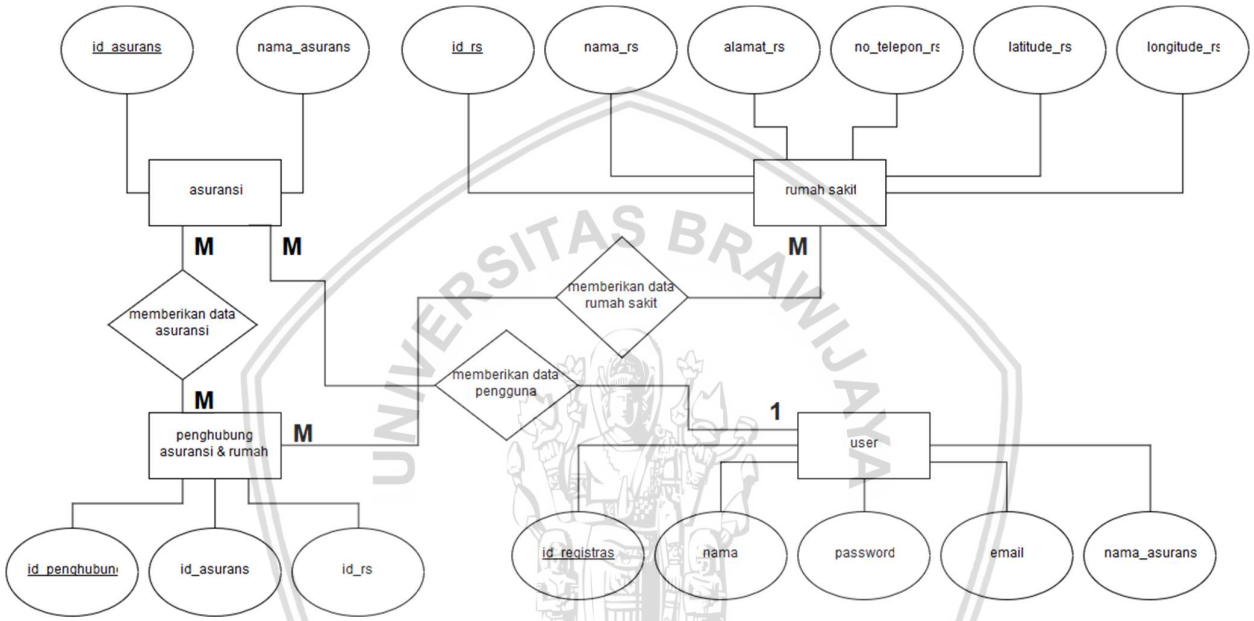

**Gambar 5.15 *Sequence Diagram* Info Rumah Sakit**

## 5.3 Perancangan Basis Data dan Komunikasi Data

Perancangan basis data diperlukan untuk menggambarkan bagaimana data yang diperlukan sistem akan disimpan. Pada penelitian ini perancangan basis data direpresentasikan dalam bentuk *ERD* (*Entity Relationship Diagram). ERD* menunjukkan hubungan yang terjadi di antara objek (*entitas)* yang terlibat dalam suatu *database*. *ERD* berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkap dengan beberapa atribut yang mempresentasikan seluruh fakta yang ditinjau dari keadaan yang nyata.

Perancangan basis data sistem ini terdapat empat buah tabel yaitu tabel asuransi kesehatan, tabel rumah sakit, tabel penghubung asuransi kesehatan dengan rumah sakit dan tabel pengguna. Tabel pertama yaitu tabel asuransi kesehatan digunakan untuk menyimpan data nama-nama asuransi kesehatan dengan id asuransi kesehatannya. Kedua tabel rumah sakit digunakan untuk menyimpan data rumah sakit yang terdiri dari id rumah sakit, nama rumah sakit, alamat rumah sakit, dan nomor telepon rumah sakit, *latitude* rumah sakit dan

*longitude* rumah sakit. Ketiga adalah tabel penghubung asuransi kesehatan dengan rumah sakit yang mana digunakan untuk menghubungkan antara data di tabel asuransi dan tabel tabel rumah sakit yang berisikan id penghubung, id asuransi dan id rumah sakit. Tabel itu akan membuat tabel baru yang berisikan id penghubung, id rumah sakit, nama rumah sakit, alamat rumah sakit, nomor telepon rumah sakit, *latitude* rumah sakit, *longitude* rumah sakit, id asuransi dan nama asuransi. Keempat adalah tabel pengguna digunakan untuk menyimpan data penguna yang terdiri dari id pengguna, nama pengguna, *password* pengguna, *email* pengguna dan asuransi kesehatan yang dimiliki pengguna.




**Gambar 5.16 ERD Sistem *Hospital Locator***

ERD dari sistem *Hospital Locator* dapat dilihat pada Gambar 5.16. Detail atribut tiap basis data dijelaskan pada Tabel 5.1.

**Tabel 5.1 Data Atribut tiap Basis Data**

| Nama Tabel | Atribut | Tipe,*Length* |
|---|---|---|
| tbl_reg | reg_id(pk) | int(11) |
| | nama | varchar(255) |
| | password | varchar(255) |
| | email | varchar(255) |
| | ak_name | varchar(255) |

**Tabel 5.1 Data Atribut tiap Basis Data (lanjutan)**

| | | |
|---|---|---|
| tbl_ak | ak_id(pk) | int(11) |
| | ak_name | varchar(255) |
| tbl_rs | rs_id(pk) | int(11) |
| | rs_name | varchar(255) |
| | rs_addr | text |
| | rs_telp | text |
| | lat | varchar(255) |
| | lon | varchar(255) |
| tbl_ak_rs | id | int(11) |
| | ak_id | int(11) |
| | rs_id | int(11) |

Perancangan komunikasi data antara aplikasi *client* dan *database* menggunakan *web service*. Proses pengiriman data menggunakan format data string dan *file* gambar yang hendak diterima oleh aplikasi *server* untuk disimpan. Proses pengiriman data dari *database* ke *client* menggunakan format data JSON, yang nantinya hendak di ubah oleh aplikasi *client* menjadi objek. Tabel 5.2 menjelaskan salah satu contoh format pertukaran data antara aplikasi *client* dan *database*.

**Tabel 5.2 Rancangan Komunikasi Data**

| **Definisi Kebutuhan** | ***Use Case*** | **Format Data** |
|---|---|---|
| Sistem menyediakan fitur bagi pengguna untuk melakukan autentifikasi *Login* dengan menginputkan *email* dan *password* | *Login* | email=”” password=”” |

**Tabel 5.2 Rancangan Komunikasi Data (lanjutan)**

| | | |
|---|---|---|
| Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | *Register* | nama=”” password=”” email=”” ak_name=”” |
| Sistem menyediakan fitur bagi pengguna untuk melihat data diri yang berisikan nama, *email, password*, dan asuransi kesehatan yang dimiliki | *View* | nama=”” password=”” email=”” ak_name=”” |
| Sistem menyediakan fitur bagi pengguna untuk mengedit data diri yang berisikan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | *Edit* | nama=”” password=”” email=”” ak_name=”” |
| Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit di Kota Malang berdasarkan prefensi asuransi kesehatan yang dimiliki oleh pengguna | Pencarian rumah sakit berdasarkan asuransi kesehatan | ak_id=”” ak_name=”” |

**Tabel 5.2 Rancangan Komunikasi Data (lanjutan)**

| | | |
|---|---|---|
| Sistem menyediakan fitur bagi pengguna untuk menampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang | Pemilihan asuransi kesehatan | ak_id=”” ak_name=”” |
| Sistem menyediakan fitur bagi pengguna untuk menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima asuransi kesehatan dalam bentuk *Maps* | Menampilkan lokasi pengguna dan lokasi rumah sakit | rs_id=”” rs_name=”” rs_addr=”” lat=”” lon=”” |
| Sistem menyediakan fitur bagi pengguna untuk menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* | Melihat detail rumah sakit | rs_id=”” rs_name=”” rs_addr=”” rs_telp=”” lat=”” lon=”” |
| Sistem menyediakan fitur bagi pengguna untuk dapat menelpon rumah sakit | Menelpon rumah sakit | rs_telp=”” |

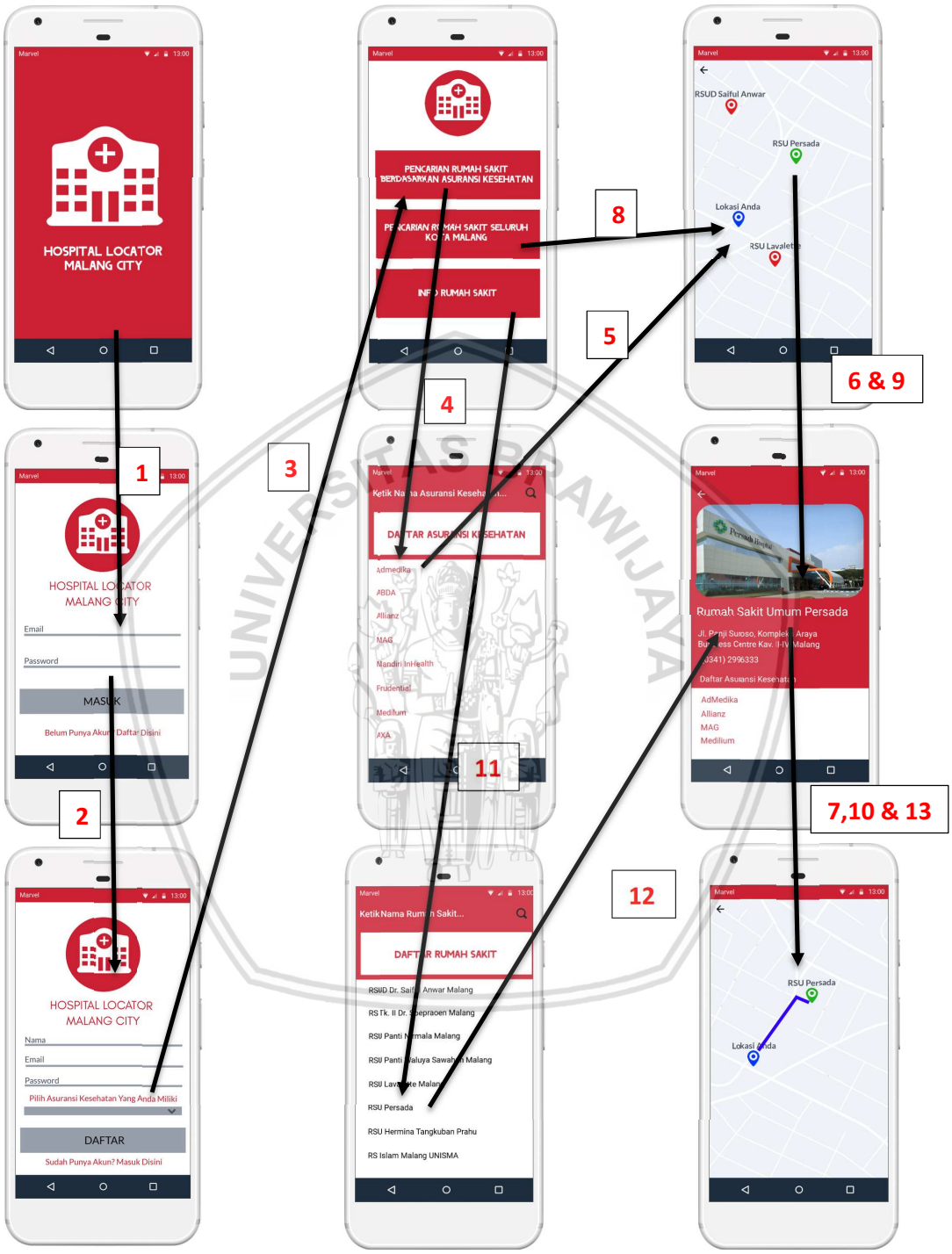
**Tabel 5.2 Rancangan Komunikasi Data (lanjutan)**

| | | |
|---|---|---|
| Sistem menyediakan fitur bagi pengguna untuk menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* | Merouting dari lokasi pengguna ke rumah sakit | rs_addr=”” lat=”” lon=”” |
| Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit seluruh Kota Malang bagi pengguna yang tidak memiliki asuransi kesehatan | Pencarian rumah sakit seluruh Kota Malang | rs_id=”” rs_name=”” rs_addr=”” lat=”” lon=”” |
| Sistem menyediakan fitur bagi pengguna untuk mengetahui info dari masing-masing rumah sakit di Kota Malang | Melihat info rumah sakit | rs_id=”” rs_name=”” |
| Sistem menyediakan fitur bagi pengguna untuk menampilkan daftar nama rumah sakit seluruh Kota Malang | Pemilihan info rumah sakit | rs_id=”” rs_name=”” |

## 5.4 Perancangan Aplikasi

Perancangan ini dilakukan untuk menentukan bagaimana antarmuka pengguna dari sistem akan dibangun dengan *screenflow*. Hasil rancangan antarmuka pengguna sistem direpresentasikan dalam bentuk denah *mockup* dan *wireframe* untuk setiap rancangan antarmuka, disertakan penjelasan mengenai komponen-komponen antarmuka yang terdapat di dalam setiap halaman tersebut.

### *5.4.1 Screenflow*




**Gambar 5.17 *Screenflow* Aplikasi Hospital Locator**

Perancangan *screenflow* ini digunakan untuk mempermudah menjelaskan alur proses antarmuka dari aplikasi pada sisi *client*.

Awalnya antarmuka akan menampilkan halaman *splash screen* yang akan memunculkan logo dari aplikasi *Hospital Locator*. Kemudian pengguna akan melakukan *login* terlebih dahulu dan bila pengguna belum memiliki akun maka pengguna harus melakukan *register* terlebih dahulu (1-2).

Kemudian akan keluar menu utama dari aplikasi *Hospital Locator* yang menunjukan tiga menu utama tentang "pencarian rumah sakit berdasarkan asuransi kesehatan", "pencarian rumah sakit seluruh Kota Malang", dan "info rumah sakit (3).

Pada halaman pencarian rumah sakit berdasarkan asuransi kesehatan awalnya akan menampilkan seluruh daftar asuransi kesehatan yang sudah disediakan oleh *database*. Setelah itu sistem akan menampilkan lokasi pengguna dan lokasi rumah sakit dalam bentuk *Maps*, kemudian pengguna dapat melihat detail rumah sakit/menelpon rumah sakit. Kemudian sistem akan menampilkan rute dengan *API Google Maps* bila telah dipilih salah satu dari rumah sakit yang tersedia (4-7).

Pada halaman pencarian rumah sakit seluruh Kota Malang sistem akan menampilkan lokasi pengguna dan lokasi rumah sakit di Kota Malang dalam bentuk *Maps*, kemudian pengguna dapat melihat detail rumah sakit/menelpon rumah sakit. Kemudian sistem akan menampilkan rute dengan *API Google Maps* bila telah dipilih salah satu dari rumah sakit yang tersedia (8-10).

Pada halaman mencari info rumah sakit awalnya akan menampilkan daftar dari rumah sakit di Kota Malang. Kemudian akan menampilkan info rumah sakit yang telah di pilih oleh pengguna yang berisikan nama, alamat, nomor telepon, daftar asuransi kesehatan yang diterima rumah sakit. Sistem juga dapat menelpon rumah sakit apabila pengguna memilih nomor telepon rumah sakit yang dipilih. Sistem juga dapat merouting dari lokasi pengguna ke rumah sakit (11-13). Gambaran *Screenflow* dari aplikasi *Hospital Locator* dapat dilihat pada Gambar 5.17.

### *5.4.2 Wireframe & Mockup UI*

*Wireframe* adalah kerangka dasar dari halaman yang akan dibangun. Secara garis besar di dalam *wireframe* ini akan menempatkan elemen-elemen penting dari halaman tersebut pada posisinya masing-masing. Secara *visual* tampilan dari *wireframe* ini hanya terdiri dari kotak dan garis yang menandakan posisi dari masing-masing elemen dari layout halaman. Sedangkan *mockup* ini berfungsi sebagai acuan kerja pembuatan aplikasi agar tidak menyimpang dari tujuan awal membuatnya. Biasanya, pembuatan aplikasi yang menggunakan *mockup* lebih efektif dan terstruktur karena pada saat pembuatan *mockup* itu sudah ditentukan kerangka pembuatan websitenya. Pada aplikasi *Hospital Locator* telah dibuat wireframe dari aplikasi dan kemudian dibuatlah *mockup UI* dari *wireframe* tersebut dan menggunakan aplikasi *Marvel*. Kemudian ditentukan warna merah sebagai warna yang dominan karena rumah sakit biasanya memakai warna tersebut.

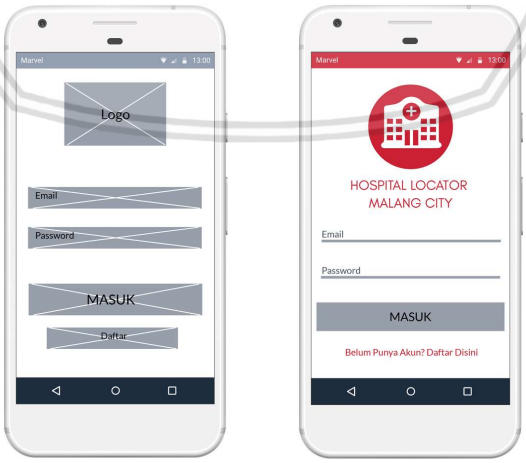
1. **Halaman *Splash Screen***

Halaman *splash screen* merupakan halaman yang tampil pertama kali saat membuka aplikasi. Pada halaman ini menampilkan logo dari aplikasi yang diberi background merah dan diberikan logo dari aplikasi *Hospital Locator*. Rancangan *Wireframe* dan *Mockup UI* Antarmuka *Splash Screen* dapat dilihat pada Gambar 5.18.


**Gambar 5.18 Rancangan *Wireframe & Mockup UI* Antarmuka *Splash Screen***

2. **Halaman *Login***

Halaman *login* merupakan halaman yang tampil pertama kali setelah tampilan *splash screen* aplikasi. Pada halaman ini menampilkan tampilan *login* yang berisikan *email* dan *password* untuk autentifikasi ke dalam aplikasi *Hospital Locator*. Rancangan *Wireframe* dan *Mockup UI* Antarmuka *Login* dapat dilihat pada Gambar 5.19.




**Gambar 5.19 Rancangan *Wireframe & Mockup UI* Antarmuka *Login***

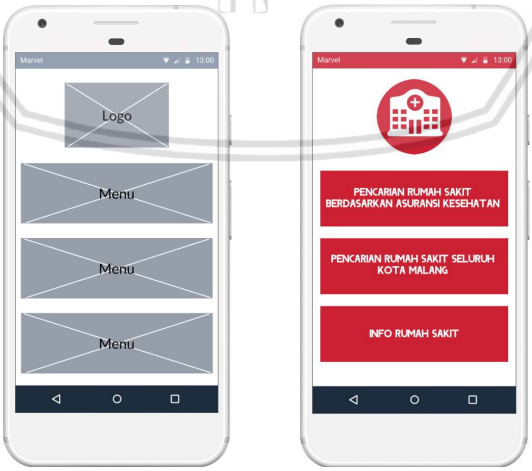
### 3. Halaman *Register*

Halaman *register* merupakan halaman untuk mendaftar ke dalam aplikasi. Pada halaman ini menampilkan halaman daftar dari halaman *login* yang berisikan nama, *email*, dan *password* untuk dapat mendaftar ke aplikasi *Hospital Locator*. Rancangan *Wireframe* dan *Mockup UI* Antarmuka *Splash Screen* dapat dilihat pada Gambar 5.20.


**Gambar 5.20 Rancangan *Wireframe & Mockup UI* Antarmuka *Register***

### 4. Halaman Menu Utama

Halaman menu utama menampilkan tiga menu utama dari aplikasi *Hospital Locator* yaitu, "Pencarian Berdasarkan Asuransi Kesehatan", "Pencarian Rumah Sakit Seluruh Kota Malang", dan "Info Rumah Sakit" yang berupa button dengan warna merah. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Menu Utama dapat dilihat pada Gambar 5.21.




**Gambar 5.21 Rancangan *Wireframe & Mockup UI* Antarmuka *Menu* Utama**

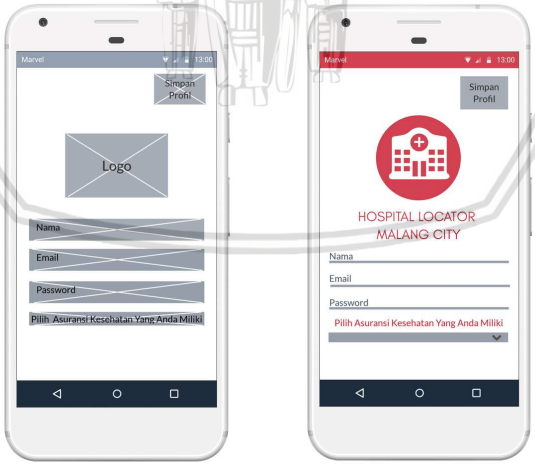
**5. Halaman *View***

Halaman *view* menampilkan data dari pengguna yang berisikan nama, email, password, dan asuransi kesehatan yang dimiliki oleh pengguna. Rancangan *Wireframe* dan *Mockup UI* Antarmuka *View* dapat dilihat pada Gambar 5.22.


**Gambar 5.22 Rancangan *Wireframe & Mockup UI* Antarmuka *View***

**6. Halaman *Edit***

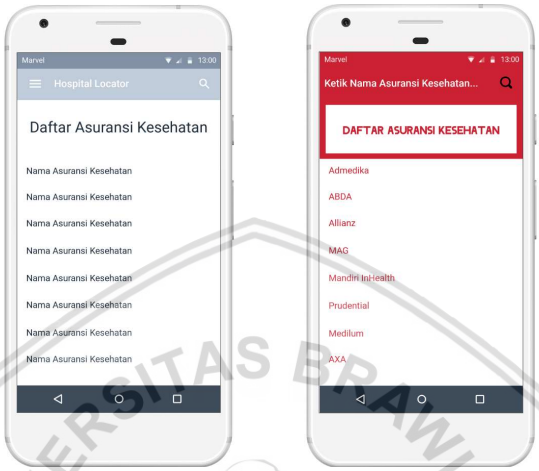
Halaman *edit* menampilkan data dari pengguna yang ingin di edit oleh pengguna berisikan nama, email, password, dan asuransi kesehatan yang dimiliki oleh pengguna. Rancangan *Wireframe* dan *Mockup UI* Antarmuka *Edit* dapat dilihat pada Gambar 5.23.




**Gambar 5.23 Rancangan *Wireframe & Mockup UI* Antarmuka *Edit***

**7. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1)**

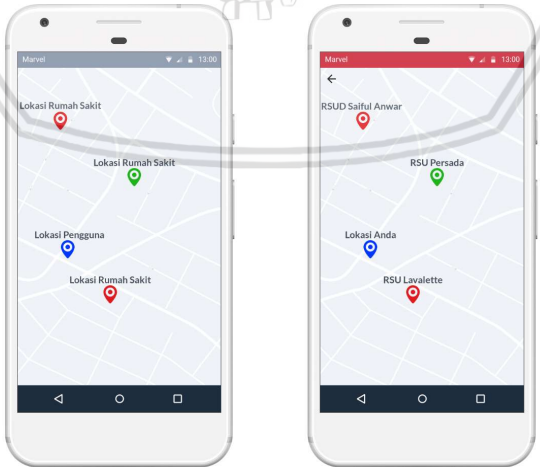
Halaman ini akan menampilkan daftar asuransi kesehatan yang tersedia di *database* dan akan menampilkannya pada pengguna asuransi kesehatan. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1) dapat dilihat pada Gambar 5.24.




**Gambar 5.24 Rancangan *Wireframe & Mockup UI* Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1)**

**8. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2)**

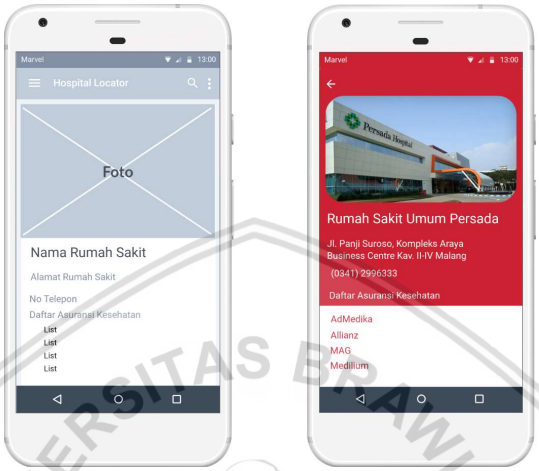
Halaman ini akan menampilkan lokasi dari pengguna dan beberapa lokasi rumah sakit yang menerima asuransi kesehatan yang sudah dipilih sebelumnya. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2) dapat dilihat pada Gambar 5.25.




**Gambar 5.25 Rancangan *Wireframe & Mockup UI* Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2)**

**9. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-3)**

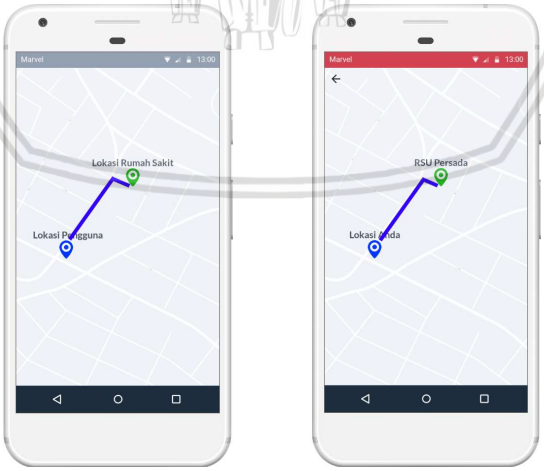
Halaman ini menampilkan info dari rumah sakit yang berisikan nama, alamat rumah sakit, no telepon, asuransi kesehatan yang diterima rumah sakit. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.26.




**Gambar 5.26 Rancangan *Wireframe & Mockup UI* Antarmuka Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-3)**

**10. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-4)**

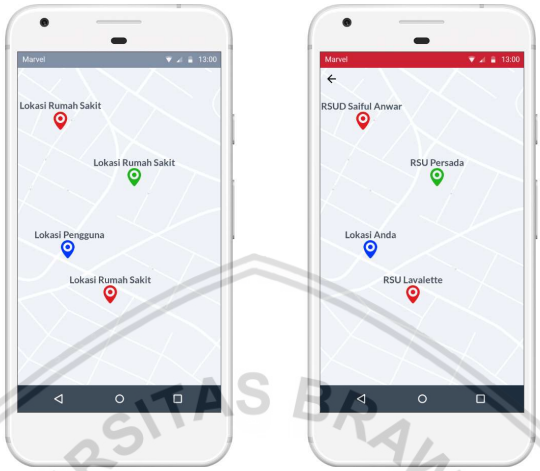
Halaman ini akan menampilkan *API Google Maps* yang akan merutekan lokasi pengguna menuju lokasi rumah sakit. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-3) dapat dilihat pada Gambar 5.27.




**Gambar 5.27 Rancangan *Wireframe & Mockup UI* Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-4)**

**11. Halaman Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-1)**

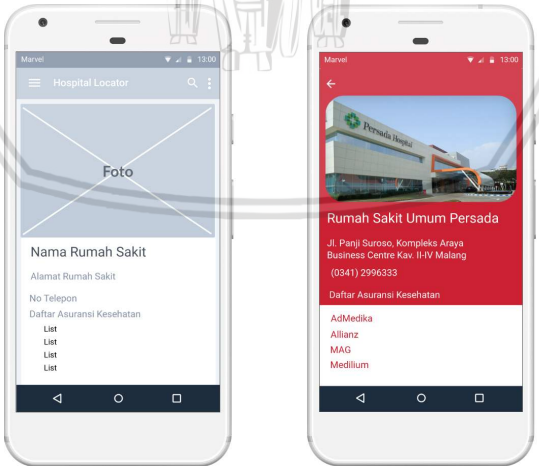
Halaman ini akan menampilkan lokasi pengguna dan lokasi seluruh rumah sakit yang ada di Kota Malang. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.28.




**Gambar 5.28 Rancangan *Wireframe & Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-1)**

**12. Halaman Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-2)**

Halaman ini menampilkan info rumah sakit yang berisikan nama, alamat, no telepon, asuransi kesehatan yang diterima rumah sakit. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.29.




**Gambar 5.29 Rancangan *Wireframe & Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-2)**

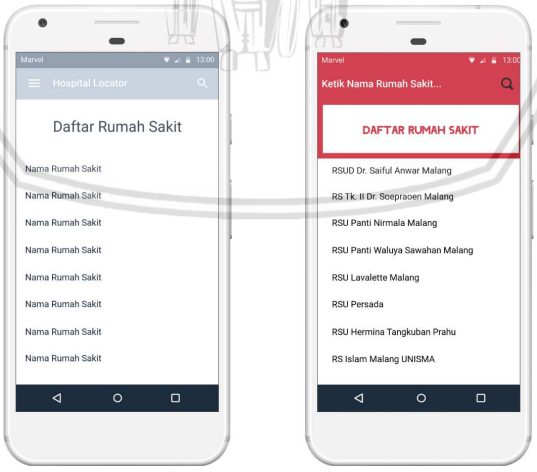
**13. Halaman Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-3)**

Halaman ini menampilkan *API Google Maps* yang merutekan lokasi pengguna menuju lokasi rumah sakit. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.30.


**Gambar 5.30 Rancangan *Wireframe & Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-3)**

**14. Halaman Info Rumah Sakit (Urutan-1)**

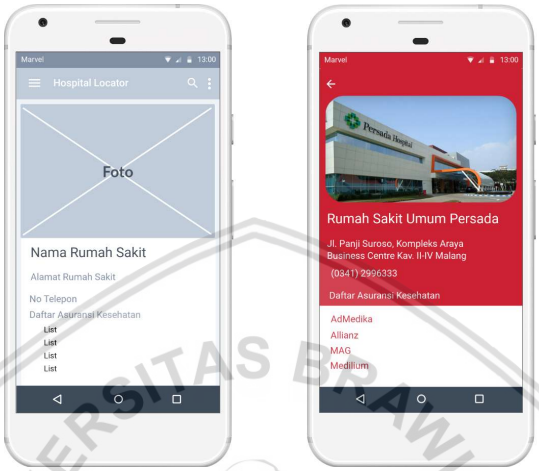
Halaman ini menampilkan daftar dari rumah sakit seluruh Kota Malang yang mana nanti pengguna akan memilih salah satu dari rumah sakit yang ingin dilihat infonya. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.31.




**Gambar 5.31 Rancangan *Wireframe & Mockup UI* Info Rumah Sakit (Ururtan-1)**

**15. Halaman Info Rumah Sakit (Urutan-2)**

Halaman ini menampilkan info dari rumah sakit yang berisikan nama, alamat rumah sakit, no telepon, asuransi kesehatan yang diterima rumah sakit. Rancangan *Wireframe* dan *Mockup UI* Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 5.32.




**Gambar 5.32 Rancangan *Wireframe & Mockup UI* Antarmuka Info Rumah Sakit (Urutan-2)**

# BAB 6 IMPLEMENTASI

Pada bab ini membahas mengenai implementasi pembuatan aplikasi pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan berasal dari hasil yang diperoleh saat melakukan analisis kebutuhan dan proses perancangan sistem. Pembahasan implementasi terdiri dari penjelasan tentang spesifikasi lingkungan implementasi, batasan-batasan implementasi, implementasi basis data, implementasi kode program, implementasi antarmuka aplikasi dan implementasi *agile system development*. Seluruh tahap tersebut diwujudkan dalam beberapa tahap: Hasil dari tahap ini menjadi jawaban atas pertanyaan penelitian pertama.

## 6.1 Spesifikasi Sistem

Hasil dari tahap analisis kebutuhan dan perancangan sistem menjadi dasar untuk dilakukan implementasi menjadi aplikasi pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan agar dapat berfungsi sesuai kebutuhan. Implementasi sistem bekerja pada lingkungan perangkat keras dan perangkat lunak. Perangkat keras yang digunakan pada lingkungan *client* adalah perangkat bergerak (*smartphone*).

### 6.1.1 Spesifikasi Perangkat Keras

Dalam pengembangan aplikasi pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan menggunakan *smartphone* Android dengan spesifikasi perangkat keras yang ditunjukkan pada Tabel 6.1.

**Tabel 6.1 Spesifikasi Perangkat Keras S*martphone* Android**

| **Nama Komponen** | **Spesifikasi** |
|---|---|
| Model Sistem | Samsung E7 |
| Prosesor | CPU Speed 1.2GHz. CPU Type Quad-Core |
| Memori Internal | 16 GB |
| Memori (RAM) | 2 GB (RAM) |
| Layar | 5.5 inches, 720 x 1280 pixels |
| WLAN | Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot |

### 6.1.2 Spesifikasi Perangkat Lunak

Dalam pengembangan aplikasi pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan mengunakan pendekatan *Location Based Service* (LBS) menggunakan Android. Proses instalasi dan pengujian perangkat dilakukan pada perangkat bergerak *smartphone* Android dengan spesifikasi perangkat lunak yang ditunjukkan pada Tabel 6.2.

**Tabel 6.2 Spesifikasi Perangkat Lunak S*martphone* Android**

| No | Nama Komponen | Spesifikasi |
|---|---|---|
| 1 | Sistem Operasi | Android Versi 5.1 (Lollipop) |

### 6.1.3 Batasan Implementasi

Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berikut:

1. Studi kasus yang digunakan dalam penelitian ini di Kota Malang.
2. Rumah sakit yang digunakan dalam penelitian ini adalah RSB. Mardi Waloeja, RSB. Permata Hati Malang, RSI. Aisyiyah, RSI. Malang UNISMA, RSIA. Galeri Candra, RSIA. Ganesha Medika, RSIA. Husada Bunda, RSIA. Melati Husada, RSIA. Mutiara Bunda Malang, RSIA. Puri Bunda Malang, RSIA. Puri Medika Malang, RSIA. Refa Husada, RST. Dr. Soepraoen, RSU. Bhakti Bunda Malang, RSU. Hermina Tangkubanprahu, RSU. Lavalette, RSU. Panti Nirmala, RSU.Panti Waluya, RSU.Permata Bunda Malang, RSU.Persada, RSU. Universitas Brawijaya, RSU. Universitas Muhammadiyah Malang, RSUD. DR. Saiful Anwar.
3. Asuransi kesehatan yang digunakan dalam penelitian ini adalah *AA Internasional, ABDA, Adira, AdMedika, AIA, Allianz, Astra Life, AVIVA, AXA, Bintang, BPJS, BRIngin Life, CAR, CHUBB, CIGNA, Common Wealth, EQUITY, FWD, Garda Medika, Great Eastern, Hanwha, I'm Care 177, Jagadiri, Jasindo, Jiwasraya, Kresna, Lippo, MAG, Mandiri InHealth, Medilum, Mega, MNC Life, NAYAKA, Pacific Cross, Pan Pacific, Panin, Prudential, Reliance, Sinarmas, SMP, SOS, Takaful, TMS*.
4. Aplikasi harus berjalan dengan terkoneksi internet sebagai media pertukaran data.
5. Untuk mendapatkan lokasi dari pengguna aplikasi dibutuhkan sensor *GPS* ataupun koneksi internet untuk mendapatkan lokasi *latitude* dan *longtitude* dari pengguna.
6. Aplikasi *Hospital Locator* dirancang untuk perangkat *mobile* dengan minimum sistem operasi berbasis Android versi 5.1 (Lollipop).
7. Implementasi peta menggunakan *API Google Maps* Android.
8. Aplikasi yang digunakan untuk pengembangan adalah Android *Studio* 2.3.3.

## 6.2 Implementasi Basis Data

Implementasi basis data perancangan perangkat lunak yang berjalan pada sisi *server* menggunakan *Database Management System* MySQL. Terdapat empat tabel pada perancangan basis data yaitu tabel *tbl_reg*, *tbl_rs, tbl_ak_rs, dan tbl_ak* yang dapat dilihat pada Gambar 6.1. Tabel-tabel tersebut dibuat sesuai dengan perancangan basis data yang dibuat berdasarkan perancangan basis data.


**Gambar 6.1 Implementasi Basis Data**

1. **Implementasi tbl_reg *(login & registrasi)***

Tabel 6.3 menjelaskan sturktur tabel *login* dan registrasi. tbl_reg digunakan untuk menyimpan data pengguna.

**Tabel 6.3 Implementasi tbl_reg**

| # | Nama | Tipe | Penyortiran | Atribut | Tak Ternilai | Bawaan | Komentar | Ekstra |
|---|---|---|---|---|---|---|---|---|
| 1 | **reg_id** | int(11) | | | Tidak | *Tidak ada* | | AUTO_INCREMENT |
| 2 | **nama** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 3 | **password** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 4 | **email** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 5 | **ak_name** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |

2. **Implementasi tbl_*ak (asuransi kesehatan)***

Tabel 6.4 menjelaskan struktur tbl_ak. tbl_ak digunakan untuk menyimpan data asuransi kesehatan.

**Tabel 6.4 Implementasi tbl_ak**

| # | Nama | Tipe | Penyortiran | Atribut | Tak Ternilai | Bawaan | Komentar | Ekstra |
|---|---|---|---|---|---|---|---|---|
| 1 | **ak_id** | int(50) | | | Tidak | *Tidak ada* | | AUTO_INCREMENT |
| 2 | **ak_name** | text | latin1_swedish_ci | | Tidak | *Tidak ada* | | |

3. **Implementasi tbl_*rs (rumah sakit)***

Tabel 6.5 menjelaskan struktur tbl_rs. tbl_rs digunakan untuk menyimpan data rumah sakit.

**Tabel 6.5 Implementasi tbl_rs**

| # | Nama | Tipe | Penyortiran | Atribut | Tak Ternilai | Bawaan | Komentar | Ekstra |
|---|---|---|---|---|---|---|---|---|
| 1 | **rs_id** | int(11) | | | Tidak | *Tidak ada* | | AUTO_INCREMENT |
| 2 | **rs_name** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 3 | **rs_addr** | text | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 4 | **rs_telp** | text | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 5 | **lat** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 6 | **lon** | varchar(255) | latin1_swedish_ci | | Tidak | *Tidak ada* | | |
| 7 | **image** | text | latin1_swedish_ci | | Tidak | *Tidak ada* | | |

**4. Implementasi tbl_*ak_rs (penghubung asuransi & rumah sakit)***

Tabel 6.6 menjelaskan struktur tbl_ak_rs. tbl_ak_rs digunakan untuk mengambil data rumah sakit berdasarkan preferensi asuransi kesehatan dan data asuransi kesehatan berdasarkan rumah sakit.

**Tabel 6.6 Implementasi tbl_ak_rs**

| # | Nama | Tipe | Penyortiran | Atribut | Tak Ternilai | Bawaan | Komentar | Ekstra |
|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | Tidak | *Tidak ada* | | AUTO_INCREMENT |
| 2 | ak_id | int(11) | | | Tidak | *Tidak ada* | | |
| 3 | rs_id | int(11) | | | Tidak | *Tidak ada* | | |

## 6.3 Implementasi Kode Program

Implementasi kode program ini merupakan tahap penulisan kode program berdasarkan fitur yang terdapat pada aplikasi pencarian rumah sakit berdasarkan preferensi asuransi kesehatan. Implemetasi kode program didapat dari hasil analisis pada perancangan *class diagram* dan *sequence diagam*. Kode program menggunakan bahasa pemrograman Java untuk aplikasi *client* dan PHP pada sisi *server*.

### 6.3.1 Kode Program Pencarian Rumah Sakit Berdasarkan Preferensi Asuransi kesehatan

Berikut ini pada Kode 6.1 adalah potongan kode program yang digunakan untuk melakukan pencarian asuransi berdasarkan asuransi kesehatan, dimana proses pendaftaran memiliki 4 tampilan untuk melakukan proses.

| No | Maps_Asuransi |
|---|---|
| 1 | private void addMarkers() { |
| 2 | if (mMap != null) |
| 3 | mMap.clear(); |
| 4 | markers = new HashMap<>(); |
| 5 | markerList = new ArrayList<>(); |
| 6 | |
| 7 | int i = 0; |
| 8 | //AK_RS |
| 9 | for(final ak_rs AK_RS : ak_rslist) { |
| 10 | |
| 11 | try { |
| 12 | if( AK_RS.lat == null \|\| AK_RS.lon == null ) |
| 13 | continue; |
| 14 | |
| 15 | if( AK_RS.lat.length() == 0 \|\| AK_RS.lon.length() == 0 ) |
| 16 | continue; |
| 17 | |
| 18 | |

```
19              if( Double.parseDouble(AK_RS.lat) == 0 ||
20  Double.parseDouble(AK_RS.lon) == 0 )
21                  continue;
22          }
23          catch(Exception e) { }
24
25          MarkerOptions markerOptionsakrs = new MarkerOptions();
26          markerOptionsakrs.title(AK_RS.rs_name);
27          markerOptionsakrs.snippet(AK_RS.rs_addr);
28
29          markerOptionsakrs.position(
30                  new LatLng(
31                          Double.parseDouble(AK_RS.lat),
32                          Double.parseDouble(AK_RS.lon)));
33
34          ak_rsHashMap.put(i++, AK_RS);
34          mMap.addMarker(markerOptionsakrs);
35          mMap.setOnInfoWindowClickListener(new
36  GoogleMap.OnInfoWindowClickListener() {
37              @Override
38              public void onInfoWindowClick(Marker marker) {
39                  for (int i = 0; i < ak_rslist.size(); i++){
40                    if
41  (ak_rsHashMap.get(i).getRs_name().equalsIgnoreCase(marker.getTitle())){
42                          Intent rv_rs = new Intent(getApplicationContext(),
43  Info_RS.class);
44                          ak_rs AK_RS = ak_rsHashMap.get(i);
46                          rv_rs.putExtra("id", AK_RS.getRs_id());
47                          rv_rs.putExtra("rs_name", AK_RS.getRs_name());
48                          rv_rs.putExtra("rs_telp", AK_RS.getRs_telp());
49                          rv_rs.putExtra("rs_addr", AK_RS.getRs_addr());
50                          rv_rs.putExtra("image", AK_RS.getImage());
51                          rv_rs.putExtra("lat", AK_RS.getLat());
52                          rv_rs.putExtra("lon", AK_RS.getLon());
53
54                          startActivity(rv_rs);
55                          return;
56                      }
57                  }
58
59              }
60          });
61
62
63      }
```

```
64      }
65
66      private void showAllPins() {
67          if(markerList == null || markerList.size() == 0)
68              return;
69
70          LatLngBounds.Builder bld = new LatLngBounds.Builder();
71          for (int i = 0; i < markerList.size(); i++) {
72              Marker marker = markerList.get(i);
73              bld.include(marker.getPosition());
74          }
75
76      }
77              private void addBoundaryToCurrentPosition(double lat, double
78  lang) {
79
80          MarkerOptions mMarkerOptions = new MarkerOptions();
81          mMarkerOptions.position(new LatLng(lat, lang));
82          mMarkerOptions.icon(BitmapDescriptorFactory
83                  .fromResource(R.drawable.map_pin));
84          mMarkerOptions.anchor(0.5f, 0.5f);
85          mMarkerOptions.title("You Are Here");
86
87          CircleOptions mOptions = new CircleOptions()
88                  .center(new LatLng(lat, lang)).radius(10000)
89                  .strokeColor(0x110000FF).strokeWidth(1).fillColor(0x110000FF);
90          mMap.addCircle(mOptions);
91          if (mCurrentPosition != null)
92              mCurrentPosition.remove();
93          mCurrentPosition = mMap.addMarker(mMarkerOptions);
        }
```

**Kode 6.1 Kode Program *Maps_Asuransi***

Kode 6.1 merupakan potongan kode program untuk menangani proses pencarian rumah sakit berdasarkan asuransi kesehatan. Penjelasan dari Kode 6.1 adalah sebagai berikut:

1. Baris 1-31 menjelaskan menunjukan lokasi rumah sakit dan pengguna menggunakan hashmap yang mana dimasukan akan mendapatkan lot dan lan dari rumah sakit yang menerima berdasarkan asuransi kesehatan dan pengguna, dan bila berhasil maka akan menampilkan *maps* yang berisikan pin dari rumah sakit yang menerima berdasarkan asuransi kesehatan dengan marker berupa nama dan alamat rumah sakit.
2. Baris 33-63 menjelaskan pengambilan detail data dari rumah sakit yang menerima berdasarkan asuransi kesehatan dengan menekan marker dari rumah sakit maka akan mengambil list dan akan mengintent ke halaman

Info_RS yang akan memanggil image, nama, alamat, nomor telpon, dan daftar asuransi rumah di terima di rumah sakit. Serta lot dan lan yang akan digunakan untuk menghubungkan dengan *API Google Maps* yang mana akan menampilkan lokasi serta nama rumah sakit yang menerima berdasarkan asuransi kesehatan dan akan dapat merouting ke rumah sakit.

3. Baris 65-93 menjelaskan pengambilan data yang telah di lakukan sebelumnya dan akan ditampilkan pin dari masing-masing rumah sakit yang menerima berdasarkan asuransi kesehatan dan menampilkan lokasi dari pengguna yang mana rumah sakit diambil berdasarkan list dan pengguna harus mengaktifkan *GPS* kemudian akan di *update* dengan lokasi pengguna berada.

### 6.3.2 Kode Program Pencarian Rumah Sakit Seluruh Kota Malang

Berikut ini pada Kode 6.2 adalah potongan kode program yang digunakan untuk melakukan proses pencarian rumah sakit seluruh Kota Malang. Dimana proses pendaftaran memiliki 3 tampilan untuk melakukan proses.

| No | Non_Asuransi |
|---|---|
| 1 | private void addMarkers() { |
| 2 | if (mMap != null) |
| 3 | mMap.clear(); |
| 4 | markers = new HashMap<>(); |
| 5 | markerList = new ArrayList<>(); |
| 6 | |
| 7 | int i = 0; |
| 8 | //RS |
| 9 | for(final rs RS : rslist) { |
| 10 | try { |
| 11 | if( RS.lat == null \|\| RS.lon == null ) |
| 12 | continue; |
| 13 | |
| 14 | if( RS.lat.length() == 0 \|\| RS.lon.length() == 0 ) |
| 15 | continue; |
| 16 | |
| 17 | if( Double.parseDouble(RS.lat) == 0 \|\| |
| 18 | Double.parseDouble(RS.lon) == 0 ) |
| 19 | continue; |
| 20 | } |
| 21 | catch(Exception e) { } |
| 22 | |
| 23 | MarkerOptions markerOptionsrs = new MarkerOptions(); |
| 24 | markerOptionsrs.title(RS.rs_name); |
| 25 | markerOptionsrs.snippet(RS.rs_addr); |
| 26 | |
| 27 | markerOptionsrs.position( |
| 28 | new LatLng( |

```
29                      Double.parseDouble(RS.lat),
30                      Double.parseDouble(RS.lon)));
31
32          Marker markrs = mMap.addMarker(markerOptionsrs);
33          markerList.add(markrs);
34          markers.put(markrs.getId(),RS);
34          mMap.setOnInfoWindowClickListener(new
35  GoogleMap.OnInfoWindowClickListener() {
36              @Override
37              public void onInfoWindowClick(Marker marker) {
38                  Intent rv_rs = new Intent(getApplicationContext(),
39  Info_RS.class);
40                  rv_rs.putExtra("id", RS.getRs_id());
41                  rv_rs.putExtra("rs_name", RS.getRs_name());
42                  rv_rs.putExtra("rs_telp", RS.getRs_telp());
43                  rv_rs.putExtra("rs_addr", RS.getRs_addr());
44                  rv_rs.putExtra("image", RS.getImage());
46                  rv_rs.putExtra("lat", RS.getLat());
47                  rv_rs.putExtra("lon", RS.getLon());
48                  startActivity(rv_rs);
49
50              }
51          });
52          Log.d("DEBUG", "addMarkersrs: LIST " +
53  markerList.get(i).toString());
54
55      }
56      Log.d("DEBUG", "addMarkersrs: LIST LENGTH " + markerList.size());
57      Log.d("DEBUG","addMarkersrs: FINISHED");
58    }
59
60    private void showAllPins() {
61      if(markerList == null || markerList.size() == 0)
62          return;
63
64      LatLngBounds.Builder bld = new LatLngBounds.Builder();
65      for (int i = 0; i < markerList.size(); i++) {
66          Marker marker = markerList.get(i);
67          bld.include(marker.getPosition());
68      }
69
70    }
71  private void addBoundaryToCurrentPosition(double lat, double lang) {
72
73      MarkerOptions mMarkerOptions = new MarkerOptions();
```

```
74        mMarkerOptions.position(new LatLng(lat, lang));
75        mMarkerOptions.icon(BitmapDescriptorFactory
76              .fromResource(R.drawable.map_pin));
77        mMarkerOptions.anchor(0.5f, 0.5f);
78        mMarkerOptions.title("You Are Here");
79
80        CircleOptions mOptions = new CircleOptions()
81              .center(new LatLng(lat, lang)).radius(10000)
82              .strokeColor(0x110000FF).strokeWidth(1).fillColor(0x110000FF);
83        mMap.addCircle(mOptions);
84        if (mCurrentPosition != null)
85           mCurrentPosition.remove();
86        mCurrentPosition = mMap.addMarker(mMarkerOptions);
87     }
```

**Kode 6.2 Kode Program *Non_Asuransi***

Kode 6.2 merupakan potongan kode program untuk menangani proses pencarian rumah sakit seluruh Kota Malang. Penjelasan dari Kode 6.2 adalah sebagai berikut:

1. Baris 1-30 menjelaskan menunjukan lokasi rumah sakit dan pengguna menggunakan hashmap yang mana dimasukan akan mendapatkan lot dan lan dari rumah sakit dan pengguna, dan bila berhasil maka akan menampilkan *maps* yang berisikan pin dari rumah sakit dengan marker berupa nama dan alamat rumah sakit.
2. Baris 33-58 menjelaskan pengambilan detail data dari rumah sakit dengan menekan marker dari rumah sakit maka akan mengambil list dan akan mengintent ke halaman Info_RS yang akan memanggil image, nama, alamat, nomor telpon, dan daftar asuransi rumah di terima di rumah sakit. Serta lot dan lan yang akan digunakan untuk menghubungkan dengan *API Google Maps* yang mana akan menampilkan lokasi serta nama rs dan akan dapat merouting ke rumah sakit.
3. Baris 60-87 menjelaskan pengambilan data yang telah di lakukan sebelumnya dan akan ditampilkan pin dari masing-masing rumah sakit dan menampilkan lokasi dari pengguna yang mana rumah sakit diambil berdasarkan list dan pengguna harus mengaktifkan *GPS* kemudian akan di *update* dengan lokasi pengguna berada.

### 6.3.3 Kode Program Info Rumah Sakit

Berikut ini pada Kode 6.3 adalah potongan kode program yang digunakan untuk melakukan proses melihat informasi dari masing-masing rumah sakit di Kota Malang. Dimana proses pendaftaran memiliki 3 tampilan untuk melakukan proses.

| No | Info_RS |
|---|---|
| 1 | protected void onCreate(Bundle savedInstanceState) { |
| 2 | |

```
3       Glide.with(getApplicationContext())
4               .load(url)
5               .apply(RequestOptions.circleCropTransform())
6               .into(im);
7
8       tv_telp.setOnClickListener(new View.OnClickListener() {
9           @Override
10          public void onClick(View view) {
11              Intent intent2 = new Intent(Intent.ACTION_DIAL);
12              intent2.setData(Uri.parse("tel:" + tv_telp.getText().toString()));
13              startActivity(intent2);
14          }
15      });
16      tv_addr.setOnClickListener(new View.OnClickListener() {
17          @Override
18          public void onClick(View view) {
19              Uri gmmIntentUri = Uri.parse("geo:" + lat + "," + lon + "?q=" + lat +
20  "," + lon + "(" + nama_rs + ")");
21              Intent mapIntent = new Intent(Intent.ACTION_VIEW,
22  gmmIntentUri);
23              mapIntent.set ("com.google.Android.apps.maps");
24              startActivity(mapIntent);
25
26          }
27      });
28
29      //query database
30      id = String.valueOf(getIntent().getIntExtra("id", 0));
31      fetchContacts();
32    }
33
34    private void fetchContacts() {
34      JsonObjectRequest request = new JsonObjectRequest(URL, new
35  JSONObject(),
36              new Response.Listener<JSONObject>() {
37                  @Override
38                  public void onResponse(JSONObject response) {
39                      if (response == null) {
40                          Toast.makeText(getApplicationContext(), "Couldn't fetch
41  the contacts! Pleas try again.", Toast.LENGTH_LONG).show();
42                          return;
43                      }
44
46                      String json = response.toString();
47                      JsonParser parser = new JsonParser();
```

```
48                  JsonElement mJson = parser.parse(json);
49
50                  Gson gson = new Gson();
51
52                  listRsAk items = gson.fromJson(mJson, listRsAk.class);
53
54                  // adding contacts to contacts list
55                  rsaklist.clear();
56
57                  StringBuilder sb = new StringBuilder();
58                  for (int i = 0; i < items.getlistRsAk().size(); i++) {
59                    if (items.getlistRsAk().get(i).getRs_id() ==
60  Integer.parseInt(id)) {
61                      sb.append(items.getlistRsAk().get(i).ak_name + ", ");
62                    }
63                  }
64                  if (sb.length() > 1)
65                    sb.delete(sb.toString().length() - 1, sb.toString().length());
66                  list_ak.setText(sb);
67
68              }
69            },
70            new Response.ErrorListener() {
71              @Override
72              public void onErrorResponse(VolleyError error) {
73
74                  Log.e(TAG, "Error: " + error.getMessage());
75                  Toast.makeText(getApplicationContext(), "Error: " +
76  error.getMessage(), Toast.LENGTH_SHORT).show();
77              }
78            });
```

**Kode 6.3 Kode Program *Info_RS***

Kode 6.3 merupakan potongan kode program untuk menangani proses info atau detail dari rumah sakit. Penjelasan dari Kode 6.3 adalah sebagai berikut:

1. Baris 1-32 menjelaskan bagaimana image dari detail terbentuk dengan menggunakan glide, memanggil *address* dari rumah sakit dengan lat dan lan dan mengintentnya ke *API Google Maps* kemudian juga mengintent call ke nomor telepon rumah sakit dengan *Action Dial.*
2. Baris 34-78 menjelaskan pengambilan data asuransi yang diterima oleh rumah sakit yang mana diambil dari list yang awalnya berbentuk JSON kemudian dibuat *stringbuilder* yang digunakan untuk menampilkan data tersebut dalam bentuk textview.

### 6.3.4 Kode Program Main Activity

Berikut ini pada Kode 6.4 adalah potongan kode program yang digunakan untuk melakukan proses mengunggah *database* dari server menuju aplikasi.

| No | Main_Activity |
|---|---|
| 1 | new JsonTaskRS().execute("http://ubdbest.com/atm/rest/data_rs.php"); |
| 2 | new JsonTaskAK().execute("http://ubdbest.com/atm/rest/data_ak.php"); |
| 3 | newJsonTaskAKRS().execute("http://ubdbest.com/atm/rest/data_ak_rs.ph |
| 4 | p"); |
| 5 | |
| 6 | public class JsonTaskRS extends AsyncTask<String, String, String> { |
| 7 | @Override |
| 8 | protected void onPreExecute() { |
| 9 | super.onPreExecute(); |
| 10 | if (db != null) { |
| 11 | q1.deleteTable("rs"); |
| 12 | Log.d("DEBUG", "onPreExecute: table deleted"); |
| 13 | } |
| 14 | } |
| 15 | |
| 16 | @Override |
| 17 | protected String doInBackground(String... params) { |
| 18 | HttpURLConnection connection = null; |
| 19 | BufferedReader reader = null; |
| 20 | |
| 21 | try { |
| 22 | URL url = new URL(params[0]); |
| 23 | connection = (HttpURLConnection) url.openConnection(); |
| 24 | connection.connect(); |
| 25 | |
| 26 | InputStream stream = connection.getInputStream(); |
| 27 | |
| 28 | reader = new BufferedReader(new InputStreamReader(stream)); |
| 29 | |
| 30 | StringBuffer buffer = new StringBuffer(); |
| 31 | Log.d("DEBUG", "doInBackground: RUNNING"); |
| 32 | String line = ""; |
| 33 | while ((line = reader.readLine()) != null) { |
| 34 | buffer.append(line); |
| 34 | |
| 35 | String finalJson = buffer.toString(); |
| 36 | |
| 37 | JSONObject parentObject = new JSONObject(finalJson); |
| 38 | JSONArray parentArray = parentObject.getJSONArray("rs"); |
| 39 | |

```
40
41              for (int i = 0; i < parentArray.length(); i++) {
42
43                  rs rsobj = new rs();
44                  JSONObject finalObject = parentArray.getJSONObject(i);
46                  rsobj.rs_id = finalObject.getInt("rs_id");
47                  rsobj.rs_name = finalObject.getString("rs_name");
48                  rsobj.rs_addr = finalObject.getString("rs_addr");
49                  rsobj.rs_telp = finalObject.getString("rs_telp");
50                  rsobj.lat = finalObject.getString("lat");
51                  rsobj.lon = finalObject.getString("lon");
52                  rsobj.image = finalObject.getString("image");
53
54
55                  q1.insertrs(rsobj);
56                  Log.d("DEBUG", "doInBackground: rsobj inserted " +
57  rsobj.rs_id);
58              }
59
60
61              return "Finish";
62          }
63
64      } catch (MalformedURLException e) {
65          e.printStackTrace();
66      } catch (IOException e) {
67          e.printStackTrace();
68      } catch (JSONException e) {
69          e.printStackTrace();
70      } finally {
71          if (connection != null) {
72              connection.disconnect();
73          }
74          try {
75              if (reader != null) {
76                  reader.close();
77              }
78          } catch (IOException e) {
79              e.printStackTrace();
80          }
81      }
82      return null;
83  }
84
85  @Override
```

```
86        protected void onPostExecute(String result) {
87           super.onPostExecute((String) result);
88           Log.d("DEBUG", "onPostExecute: " + result);
89        }
90
91        @Override
92        protected void finalize() throws Throwable {
93           super.finalize();
94        }
95     }
96
97  public class JsonTaskAK extends AsyncTask<String, String, String> {
98        @Override
99        protected void onPreExecute() {
100          super.onPreExecute();
101          if (db != null) {
102             q2.deleteTable("ak");
103             Log.d("DEBUG", "onPreExecute: table deleted");
104          }
105       }
106
107       @Override
108       protected String doInBackground(String... params) {
109          HttpURLConnection connection = null;
110          BufferedReader reader = null;
111
112          try {
113             URL url = new URL(params[0]);
114             connection = (HttpURLConnection) url.openConnection();
115             connection.connect();
116
117             InputStream stream = connection.getInputStream();
118
119             reader = new BufferedReader(new InputStreamReader(stream));
120
121             StringBuffer buffer = new StringBuffer();
122             Log.d("DEBUG", "doInBackground: RUNNING");
123             String line = "";
124             while ((line = reader.readLine()) != null) {
125                buffer.append(line);
126
127                String finalJson = buffer.toString();
128
129                JSONObject parentObject = new JSONObject(finalJson);
130                JSONArray parentArray = parentObject.getJSONArray("ak");
```

```
131
132
133             for (int i = 0; i < parentArray.length(); i++) {
134
135                 ak akobj = new ak();
136                 JSONObject finalObject = parentArray.getJSONObject(i);
137                 akobj.ak_id = finalObject.getInt("ak_id");
138                 akobj.ak_name = finalObject.getString("ak_name");
139
140
141
142                 q2.insertak(akobj);
143                 Log.d("DEBUG", "doInBackground: akobj inserted " +
144 akobj.ak_id);
145             }
146
147
148             return "Finish";
149         }
150
151       } catch (MalformedURLException e) {
152         e.printStackTrace();
153       } catch (IOException e) {
154         e.printStackTrace();
155       } catch (JSONException e) {
156         e.printStackTrace();
157       } finally {
158         if (connection != null) {
159           connection.disconnect();
160         }
161         try {
162           if (reader != null) {
163             reader.close();
164           }
165         } catch (IOException e) {
166           e.printStackTrace();
167         }
168       }
169       return null;
170     }
171
172     @Override
173     protected void onPostExecute(String result) {
174       super.onPostExecute((String) result);
175       Log.d("DEBUG", "onPostExecute: " + result);
```

```
176         }
177
178         @Override
179         protected void finalize() throws Throwable {
180             super.finalize();
181         }
182     }
183
184     public class JsonTaskAKRS extends AsyncTask<String, String, String> {
185         @Override
186         protected void onPreExecute() {
187             super.onPreExecute();
188             if (db != null) {
189                 q3.deleteTable("ak_rs");
190                 Log.d("DEBUG", "onPreExecute: table deleted");
191             }
192         }
193
194         @Override
195         protected String doInBackground(String... params) {
196             HttpURLConnection connection = null;
197             BufferedReader reader = null;
198
199             try {
200                 URL url = new URL(params[0]);
201                 connection = (HttpURLConnection) url.openConnection();
202                 connection.connect();
203
204                 InputStream stream = connection.getInputStream();
205
206                 reader = new BufferedReader(new InputStreamReader(stream));
207
208                 StringBuffer buffer = new StringBuffer();
209                 Log.d("DEBUG", "doInBackground: RUNNING");
210                 String line = "";
211                 while ((line = reader.readLine()) != null) {
212                     buffer.append(line);
213
214                     String finalJson = buffer.toString();
215
216                     JSONObject parentObject = new JSONObject(finalJson);
217                     JSONArray parentArray = parentObject.getJSONArray("ak_rs");
218
219
220                     for (int i = 0; i < parentArray.length(); i++) {
```

```
221
222                 ak_rs akrsobj = new ak_rs();
223                 JSONObject finalObject = parentArray.getJSONObject(i);
224                 akrsobj.id = finalObject.getInt("id");
225                 akrsobj.ak_id = finalObject.getInt("ak_id");
226                 akrsobj.rs_id = finalObject.getInt("rs_id");
227                 akrsobj.ak_name = finalObject.getString("ak_name");
228                 akrsobj.rs_name = finalObject.getString("rs_name");
229                 akrsobj.rs_addr = finalObject.getString("rs_addr");
230                 akrsobj.lat = finalObject.getString("lat");
231                 akrsobj.lon = finalObject.getString("lon");
232                 akrsobj.image = finalObject.getString("image");
233                 akrsobj.rs_telp = finalObject.getString("rs_telp");
234
235
236                 q3.insertak_rs(akrsobj);
237                 Log.d("DEBUG", "doInBackground: akrsobj inserted " +
238 akrsobj.id);
239             }
240
241
242             return "Finish";
243         }
244
245       } catch (MalformedURLException e) {
246         e.printStackTrace();
247       } catch (IOException e) {
248         e.printStackTrace();
249       } catch (JSONException e) {
250         e.printStackTrace();
251       } finally {
252         if (connection != null) {
253           connection.disconnect();
254         }
255         try {
256           if (reader != null) {
257             reader.close();
258           }
259         } catch (IOException e) {
260           e.printStackTrace();
261         }
262       }
263       return null;
264     }
265
```

```
266         @Override
267         protected void onPostExecute(String result) {
268             super.onPostExecute((String) result);
269             Log.d("DEBUG", "onPostExecute: " + result);
270         }
271
272         @Override
273         protected void finalize() throws Throwable {
274             super.finalize();
275         }
276     }
```

**Kode 6.4 Kode Program *Main_Activity***

Kode 6.4 merupakan potongan kode program untuk menangani proses *main activity* yang mana digunakan untuk pengambilan data. Penjelasan dari Kode 6.4 adalah sebagai berikut:

1. Baris 1-4 menjelaskan pengambilan data dari server dengan *JsonTask.*
2. Baris 6-95 menjelaskan pengambilan *JsonTaskRS* kemudian akan dibuat ke dalam URL yang mana akan dibuat object yang akan diisikan *rs_id, rs_name, rs_addr, rs_telp, lat, lon,* dan *image* kemudian akan dimasukan kedalam database aplikasi secara *background.*
3. Baris 97-182 menjelaskan pengambilan *JsonTaskAK* kemudian akan dibuat ke dalam URL yang mana akan dibuat object yang akan diisikan *ak_id* dan *ak_name* kemudian akan dimasukan kedalam database aplikasi secara *background.*
4. Baris 184-276 menjelaskan pengambilan *JsonTaskAKRS* kemudian akan dibuat ke dalam URL yang mana akan dibuat object yang akan diisikan *id, ak_id, rs_id, ak_name, rs_name, rs_addr, lat, lon, image* dan *rs_telp* kemudian akan dimasukan kedalam database aplikasi secara *background.*

## 6.4 Implementasi Antarmuka

Pada tahap ini dijelaskan mengenai berbagai hasil implementasi antarmuka pengguna yang telah diimplementasikan berdasarkan *wireframe* dan *mockup UI* yang telah dilakukan.

### 1. Halaman *Splash Screen*

Halaman *splash screen* merupakan halaman yang tampil pertama kali saat membuka aplikasi. Pada halaman ini menampilkan logo dari aplikasi *Hospital Locator*. Implementasi Antarmuka *Splash Screen* dapat dilihat pada Gambar 6.2.

**2. Halaman *Login***

Halaman *login* merupakan halaman yang tampil pertama kali setelah tampilan *splash screen* aplikasi. Pada halaman ini menampilkan tampilan *login* yang berisikan *email* dan *password* untuk autentifikasi ke dalam aplikasi *Hospital Locator*. Implementasi Antarmuka *Login* dapat dilihat pada Gambar 6.2.

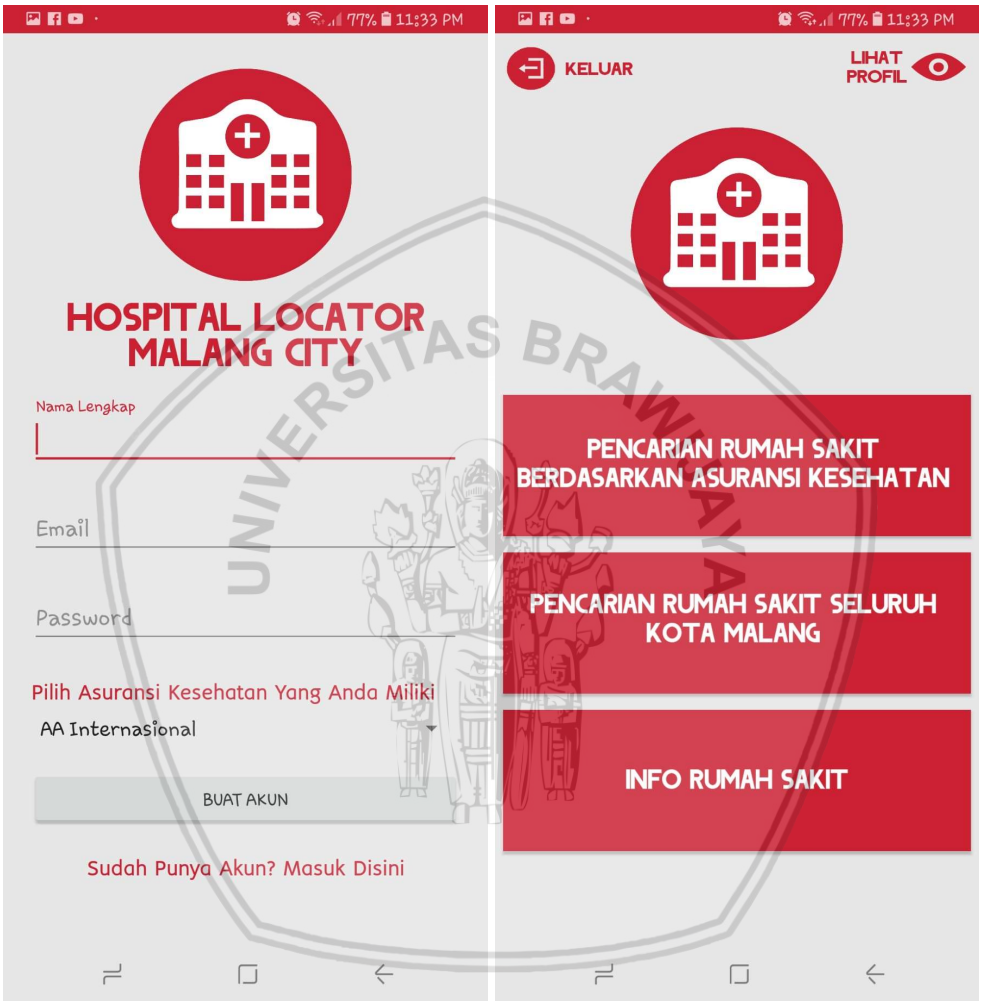

**Gambar 6.2 Implementasi *Splash Screen* dan *Login***

**3. Halaman *Register***

Halaman *register* merupakan halaman untuk mendaftar ke dalam aplikasi. Pada halaman ini menampilkan halaman daftar dari halaman *login* yang berisikan nama, *email*, dan *password* untuk dapat mendaftar ke aplikasi *Hospital Locator*. Implementasi Antarmuka *Register* dapat dilihat pada Gambar 6.3.

**4. Halaman Menu Utama**

Halaman menu utama menampilkan tiga menu utama dari aplikasi *Hospital Locator* yaitu, "Pencarian Berdasarkan Asuransi Kesehatan", "Pencarian Rumah Sakit Seluruh Kota Malang", dan "Info Rumah Sakit" yang berupa button dengan warna merah. Implementasi Antarmuka Menu Utama dapat dilihat pada Gambar 6.3.




**Gambar 6.3 Implementasi *Registrasi* dan Menu Utama**

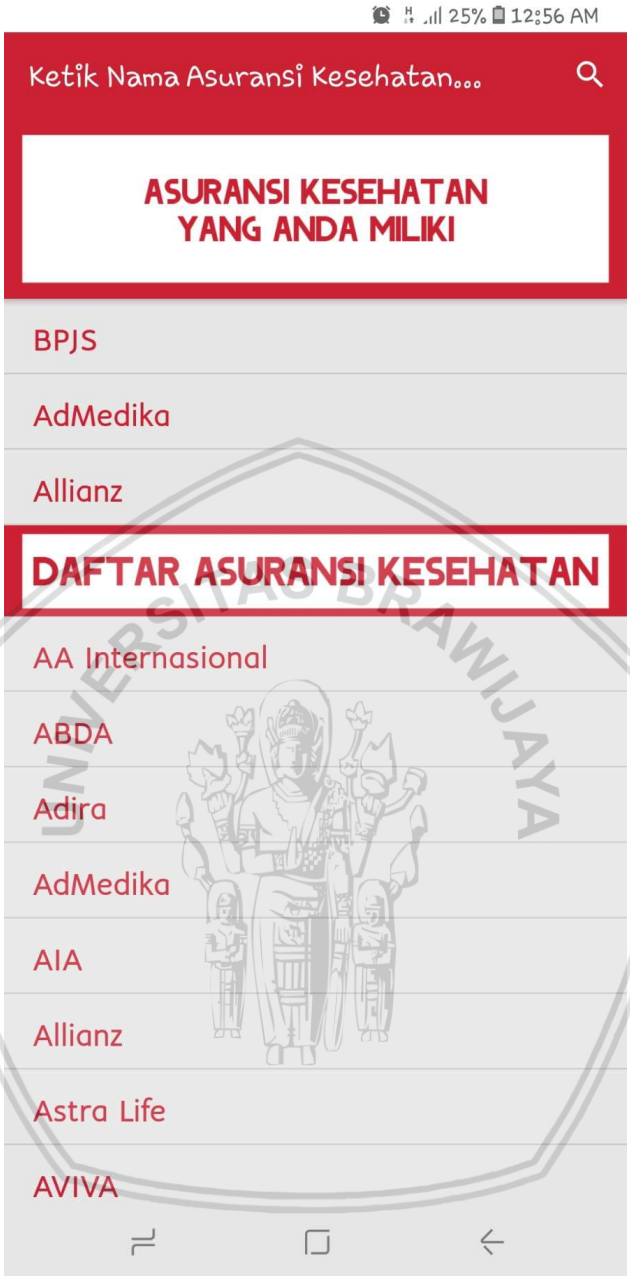
**5. Halaman *View & Edit***

Halaman ini akan menampilkan data dari pengguna yang berisikan nama, email, password, dan daftar asuransi yang dimiliki oleh pengguna. Kemudian pengguna dapat mengedit datanya dengan memasukan inputan yang sama untuk merubah data dirinya. Implementasi Antarmuka *View & Edit* dapat dilihat pada Gambar 6.4.


**Gambar 6.4 Implementasi *View & Edit***

**6. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1)**

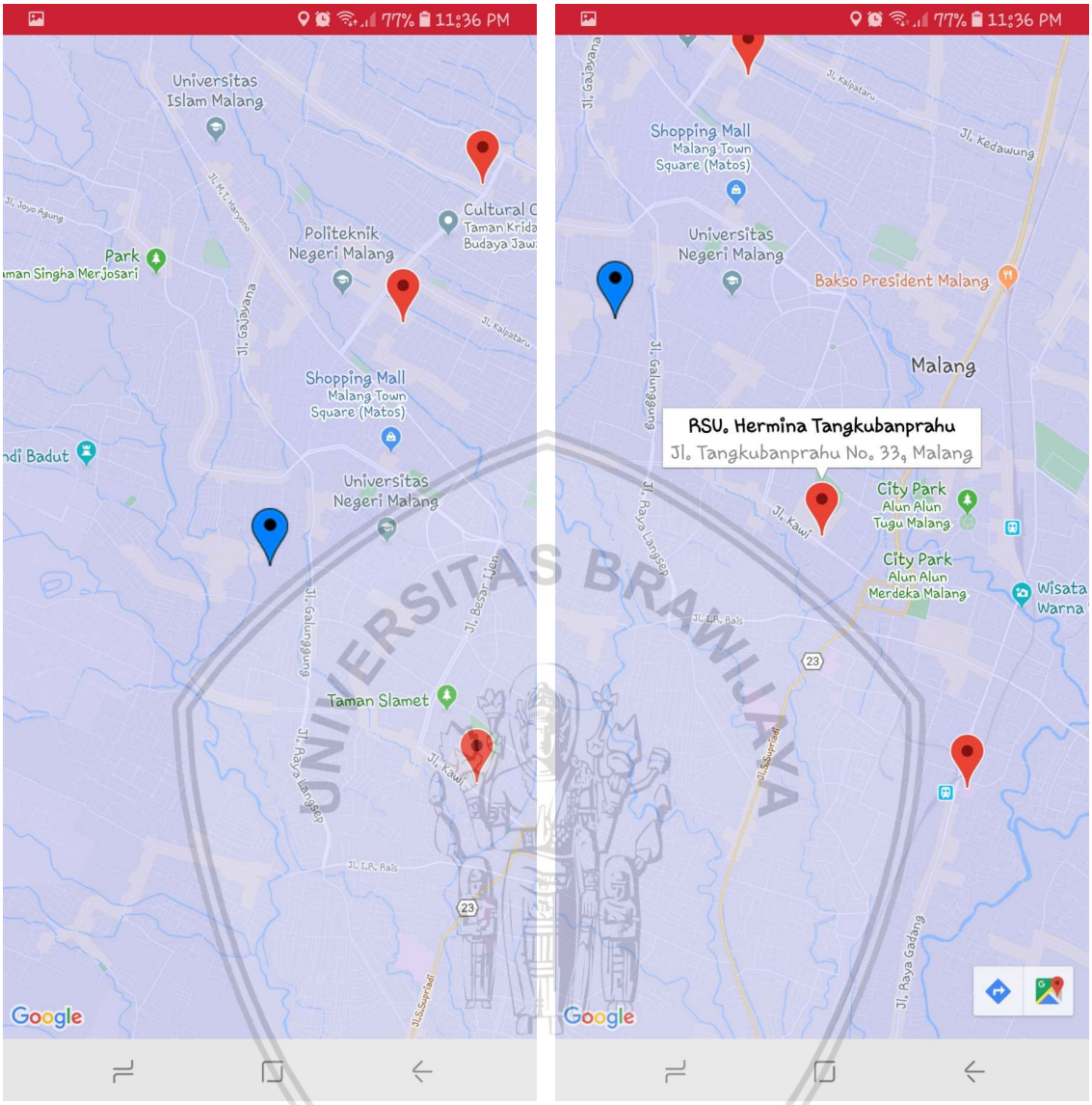
Halaman ini akan menampilkan daftar asuransi kesehatan yang tersedia di database dan akan menampilkannya pada pengguna asuransi kesehatan. Implementasi Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1) dapat dilihat pada Gambar 6.5.




**Gambar 6.5 Implementasi Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-1)**

**7. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2)**

Halaman ini akan menampilkan lokasi dari pengguna dan beberapa lokasi rumah sakit yang menerima asuransi kesehatan yang sudah dipilih sebelumnya. Implementasi Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2) dapat dilihat pada Gambar 6.6.




**Gambar 6.6 Implementasi Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2)**

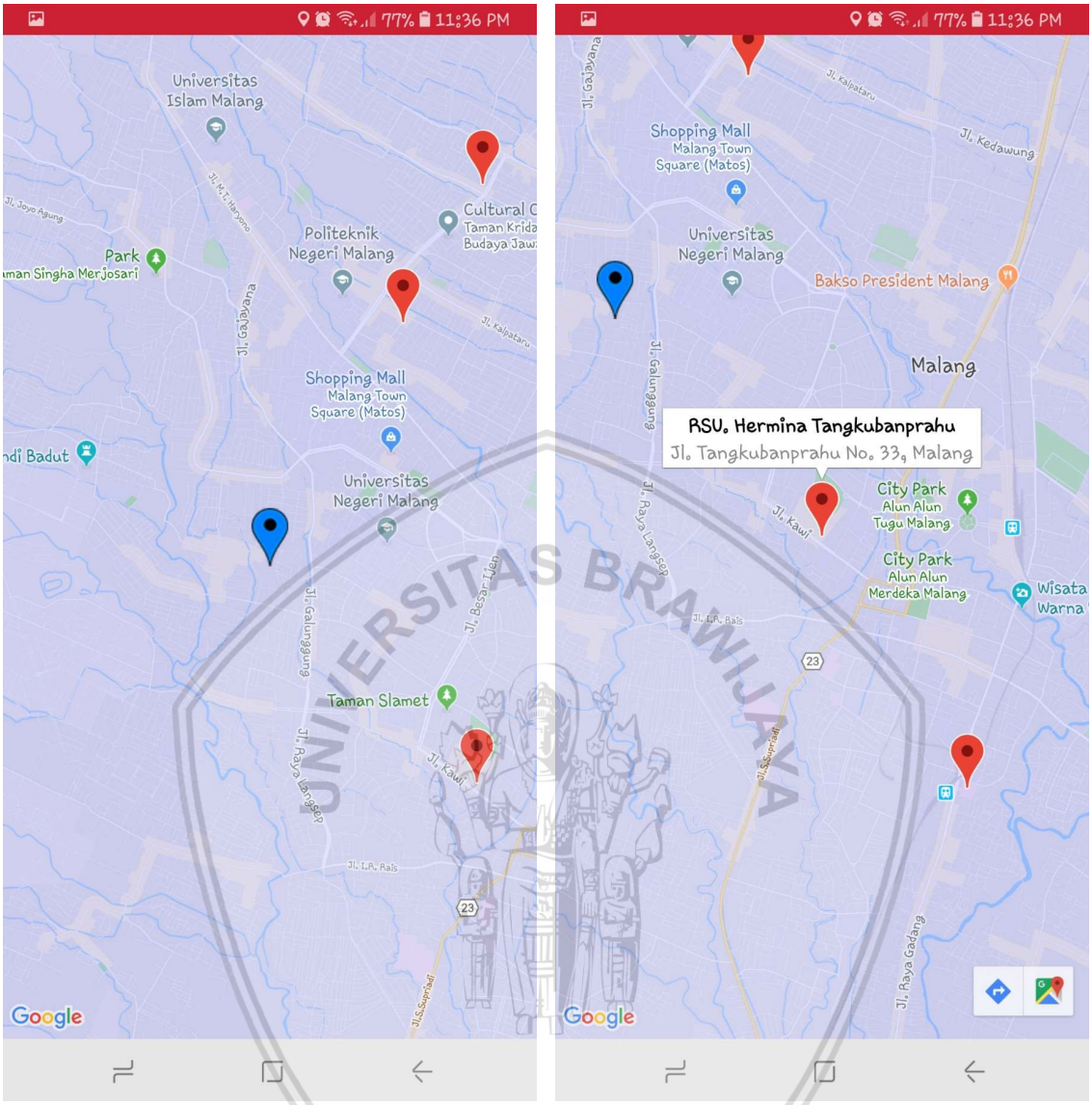
**8. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-3)**

Halaman ini menampilkan info dari rumah sakit yang berisikan nama, alamat rumah sakit, no telepon, asuransi kesehatan yang diterima rumah sakit. Implementasi Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 6.7.


**Gambar 6.7 Implementasi Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-3)**

**9. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-4)**

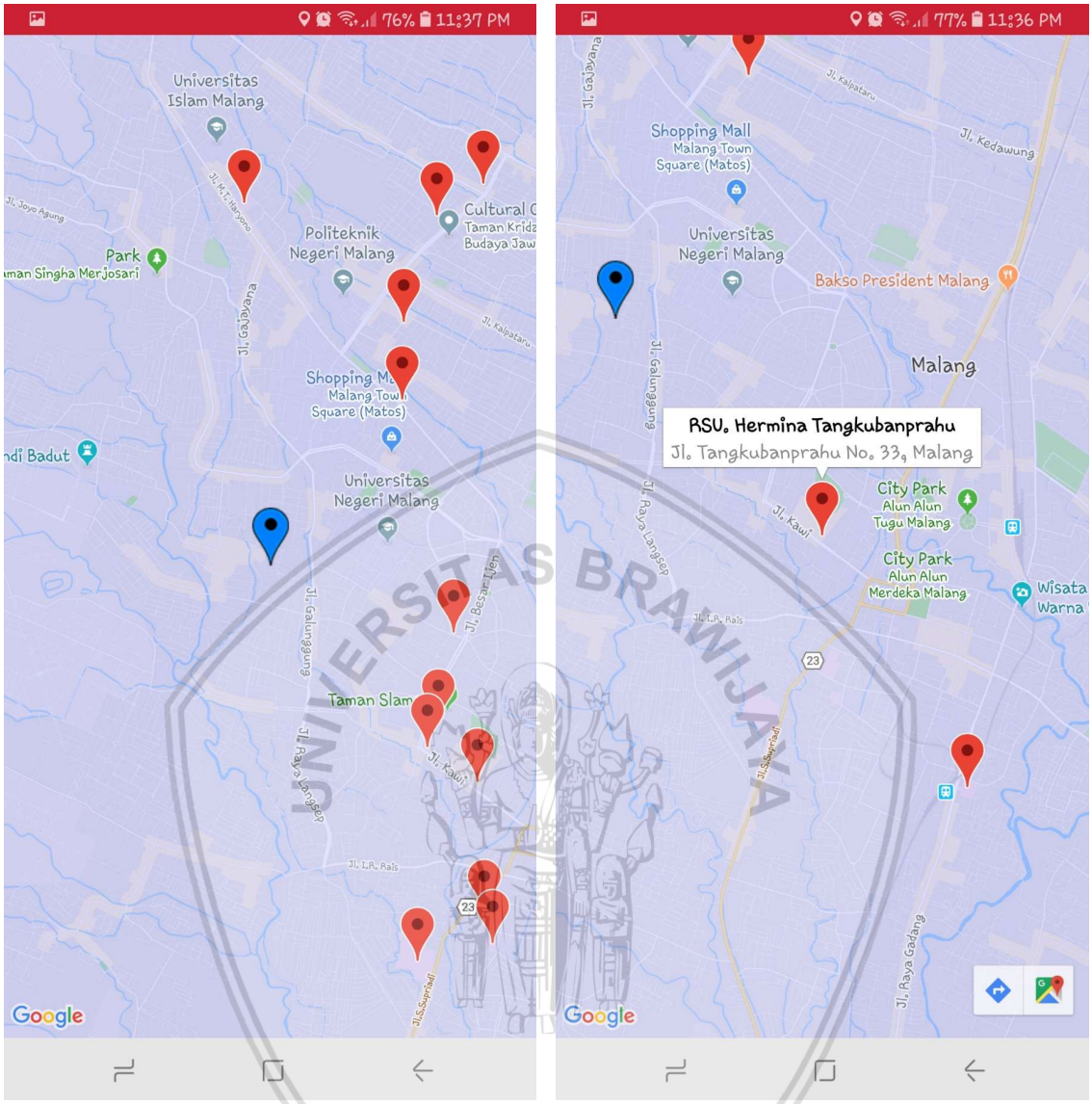
Halaman ini akan menampilkan *API Google Maps* yang akan merutekan lokasi pengguna menuju lokasi rumah sakit. Implementasi Antarmuka Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-4) dapat dilihat pada Gambar 6.8.




**Gambar 6.8 Implementasi Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-4)**

### 10. Halaman Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-1)

Halaman ini akan menampilkan lokasi pengguna dan lokasi seluruh rumah sakit yang ada di Kota Malang. Implementasi Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 6.9.




**Gambar 6.9 Implementasi Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-1)**

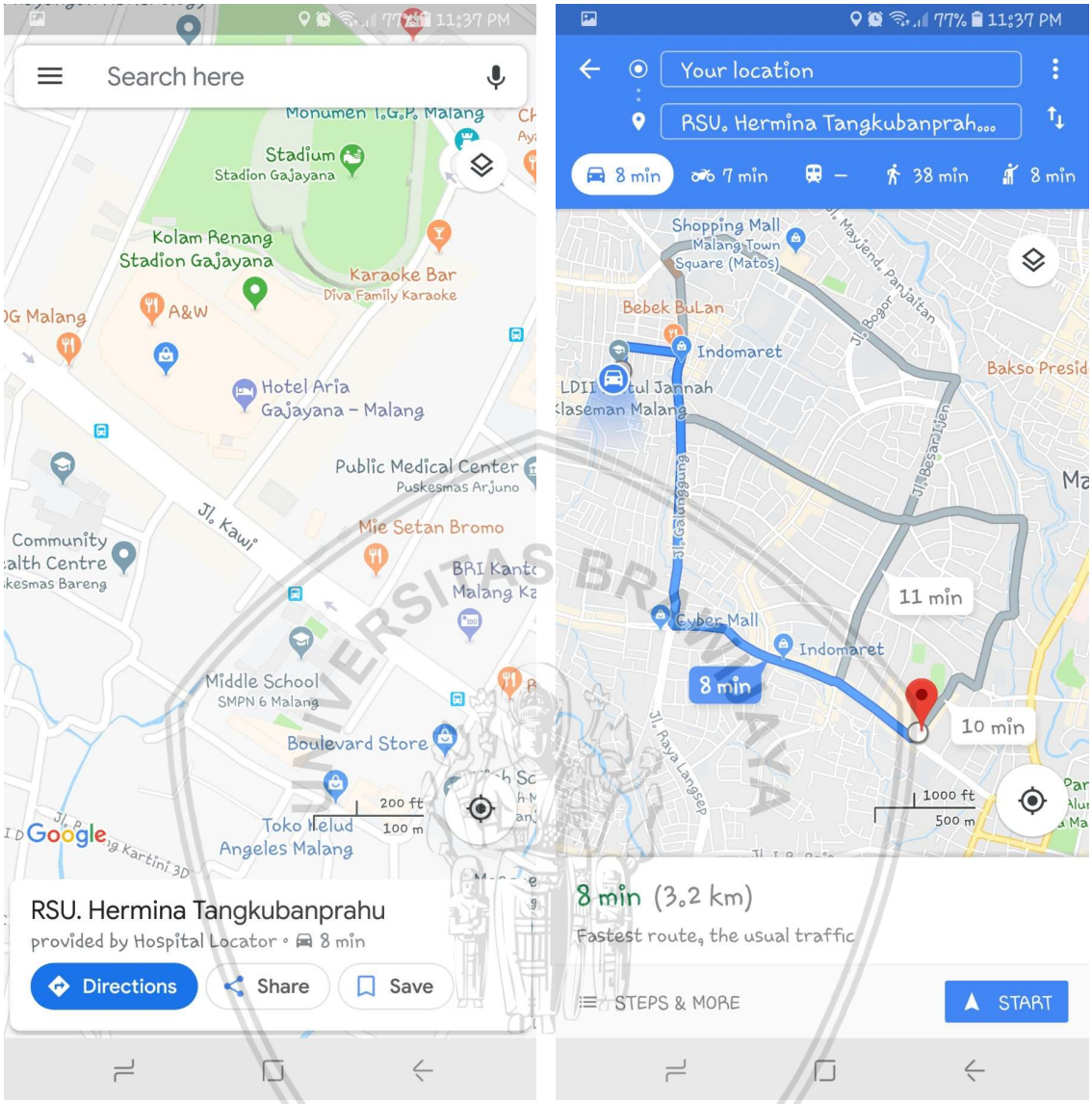
### 11. Halaman Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (Urutan-2)

Halaman ini menampilkan info dari rumah sakit yang berisikan nama, alamat rumah sakit, no telepon, asuransi kesehatan yang diterima rumah sakit. Implementasi Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 6.10.


**Gambar 6.10 Implementasi Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-2)**

**12. Halaman Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-3)**

Halaman ini akan menampilkan *API Google Maps* yang akan merutekan lokasi pengguna menuju lokasi rumah sakit. Implementasi Antarmuka Pencarian Rumah Sakit Seluruh Kota Malang dapat dilihat pada Gambar 6.11.



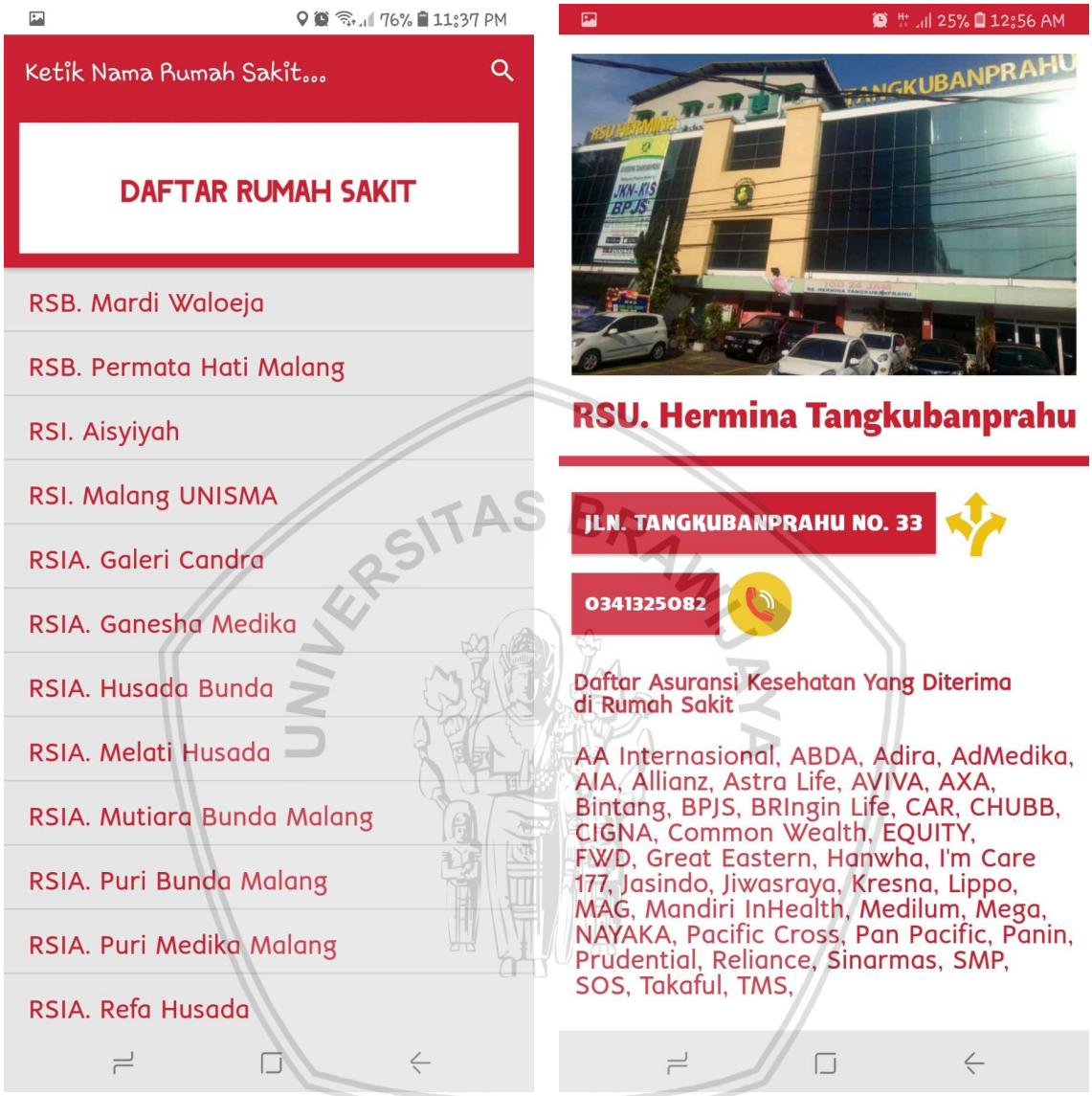

**Gambar 6.11 Implementasi Pencarian Rumah Sakit Seluruh Kota Malang (Urutan-3)**

**13. Halaman Info Rumah Sakit (Urutan-1)**

Halaman ini menampilkan daftar dari rumah sakit seluruh Kota Malang yang disediakan oleh database dan akan menampilkannya pada pengguna. Implementasi Antarmuka Info Rumah Sakit dapat dilihat pada Gambar 6.12.

**14. Halaman Info Rumah Sakit (Urutan-2)**

Halaman ini menampilkan info dari rumah sakit yang berisikan nama, alamat rumah sakit, no telepon, asuransi kesehatan yang diterima rumah sakit. Implementasi Antarmuka Info Rumah Sakit dapat dilihat pada Gambar 6.12.



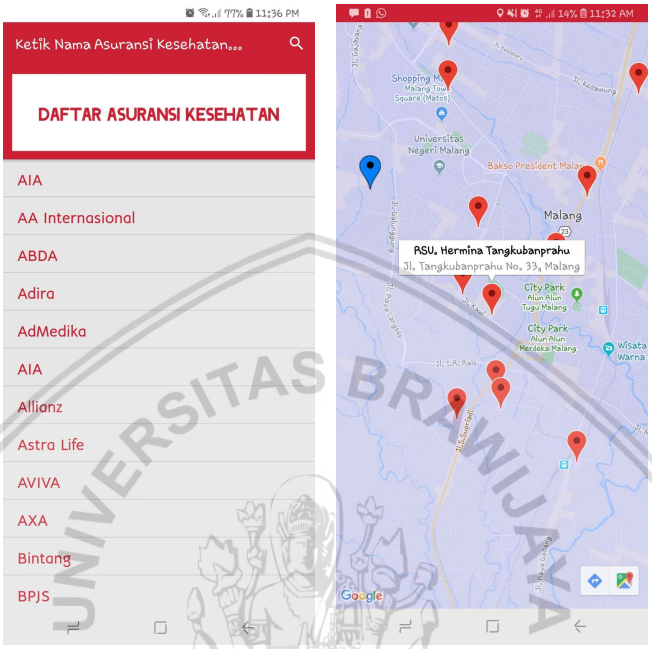

**Gambar 6.12 Implementasi Info Rumah Sakit (Urutan-1) & (Urutan-2)**

## *6.5* Implementasi *Agile System Development*

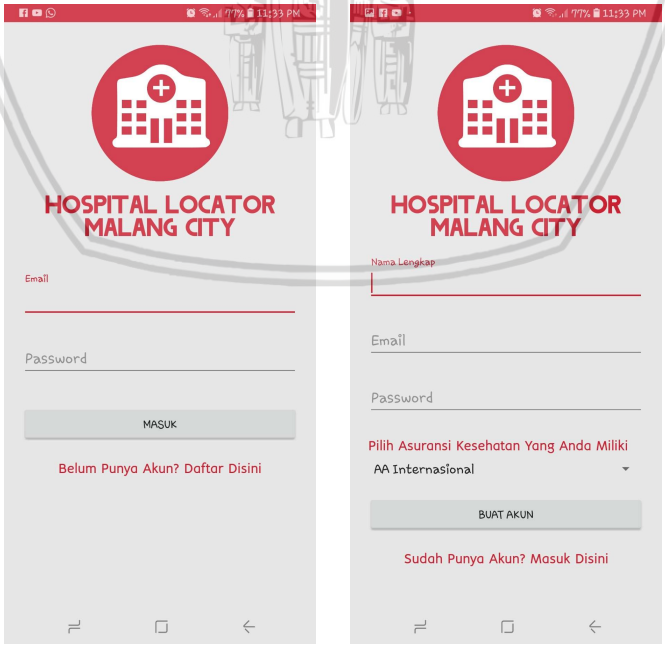
Pada Iterasi pertama yang melibatkan 10 responden, aplikasi awalnya hanya dapat mencari rumah sakit di Kota Malang berdasarkan asuransi kesehatan yang dimiliki oleh pengguna. Kemudian pengguna dapat melihat seluruh rumah sakit di Kota Malang, melihat info dari masing-masing rumah sakit yang ada di Kota Malang dan merouting dari lokasi pengguna ke lokasi rumah sakit yang dapat dilihat pada Gambar 6.13.

Pada iterasi kedua yang melibatkan 20 responden, aplikasi mendapatkan kebutuhan fungsional yang baru dengan menambahkan fungsi yang mana pengguna dapat mendaftarkan dirinya yang berisikan nama, password, email dan

asuransi yang dipunya oleh pengguna yang mana saat pengguna ingin mencari asuransi yang dimilikinya tidak merasa kesulitan. Kemudian menambahkan fitur melihat dan menyunting data dirinya dan dapat menelpon rumah sakit bilamana sewaktu-waktu pengguna dalam keadaan darurat dan membutuhkan bantuan ambulance dari rumah sakit sakit yang dapat dilihat pada Gambar 6.14.




**Gambar 6.13 Implementasi Iterasi Pertama *Agile System Development***




**Gambar 6.14 Implementasi Iterasi Kedua *Agile System Development***


**Gambar 6.14 Implementasi Iterasi Kedua *Agile System Development* (lanjutan)**

# BAB 7 PENGUJIAN

Pada bab ini membahas mengenai tahapan pengujian serta analisis perangkat bergerak aplikasi *mobile* untuk pencarian rumah sakit di Kota Malang berdasarkan preferensi asuransi kesehatan. Pengujian dilakukan dengan dua metode, yaitu *validity testing* dan *usability testing*. Seluruh tahap tersebut diwujudkan dalam beberapa tahap: Hasil dari tahap ini menjadi jawaban atas pertanyaan penelitian kedua.

## 7.1 Pengujian

Pengujian dilakukan untuk dapat memastikan hasil implementasi dapat berjalan sesuai dengan yang diharapkan dari perancangan yang telah dibuat, termasuk menemukan kesalahan dan berbagai macam kemungkinan yang dapat menimbulkan kesalahan sesuai dengan spesifikasi perangkat bergerak yang telah ditentukan sebelum sistem diserahkan kepada *client*. Pengujian menunjukan bahwa perangkat lunak dapat bekerja sesuai spesifikasi dan persayaratan kinerja telah dipenuhi. Pada aplikasi yang dibuat dalam penelitian ini dilakukan pengujian berupa *validity testing* dan *usability testing*.

### 7.1.1 *Validity Testing*

*Validity testing* ini ditujukan untuk mengetahui apakah setiap fungsional dari aplikasi yang dibangun sudah sesuai dengan kebutuhan yang telah dirumuskan dan merupakan hasil analisis kebutuhan. *Validity testing* menggunakan pengujian *blackbox* karena tidak diperlukan fokus terhadap alur jalannya kode program melainkan lebih difokuskan untuk menemukan kesalahan antara kinerja aplikasi dengan daftar kebutuhan. Setiap kasus uji yang dibuat memiliki ID kasus uji, nama kasus uji, ID *use case*, tujuan pengujian, prosedur pengujian dan hasil yang diharapkan. Kasus uji kebutuhan fungsional yang menjadi fokus utaman dapat dilihat pada Tabel 4.3 hingga Tabel 4.14.

**Tabel 7.1 Kasus *Validity Testing Login***

| | |
|---|---|
| **ID KASUS UJI** | **PV-001** |
| **NAMA KASUS UJI** | *Login* |
| **ID *USE CASE*** | FRS-100 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk melakukan autentifikasi *Login* dengan menginputkan *email* dan *password* |
| **PROSEDUR PENGUJIAN** | 1. Aktor dapat memulai *login* dengan menginputkan *email* dan *password* yang mana aktor telah mendaftarkan sebelumnya |

**Tabel 7.1 Kasus Validity Testing Login (lanjutan)**

| | |
|---|---|
| **PROSEDUR PENGUJIAN** | 2. Aktor menekan tombol masuk untuk menjalankan aplikasi *Hospital Locator*<br>3. Bila aktor belum memiliki akun maka aktor harus melakukan registrasi terlebih dahulu<br>4. Bila aktor menginputkan *email*/*password* yang salah maka tidak akan masuk ke dalam aplikasi *Hospital Locator* dan akan muncul pemberitahuan |

**Tabel 7.2 Kasus *Validity Testing Register***

| | |
|---|---|
| **ID KASUS UJI** | **PV-002** |
| **NAMA KASUS UJI** | *Register* |
| **ID *USE CASE*** | FRS-101 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki |
| **PROSEDUR PENGUJIAN** | 5. Aktor menekan tombol daftar disini<br>6. Aktor menginputkan nama, *email*, *password* dan asuransi kesehatan yang dimiliki oleh aktor<br>7. Aktor menekan tombol daftar dan akan masuk kembali ke halaman *login* system<br>8. Aktor mengisikan *email* dan *password* untuk masuk ke dalam aplikasi *Hospital Locator* |

**Tabel 7.3 Kasus *Validity Testing View***

| | |
|---|---|
| **ID KASUS UJI** | **PV-003** |
| **NAMA KASUS UJI** | *View* |
| **ID *USE CASE*** | FRS-102 |

**Tabel 7.3 Kasus Validity Testing View (lanjutan)**

| | |
|---|---|
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk melihat data diri yang berisikan nama, *email, password*, dan asuransi kesehatan yang dimiliki |
| **PROSEDUR PENGUJIAN** | 1. Aktor menekan tombol lihat profil<br>2. Aktor diperlihatkan nama, *email*, *password* dan asuransi kesehatan yang dimiliki oleh aktor |

**Tabel 7.4 Kasus *Validity Testing Edit***

| | |
|---|---|
| **ID KASUS UJI** | **PV-004** |
| **NAMA KASUS UJI** | *Edit* |
| **ID *USE CASE*** | FRS-103 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki |
| **PROSEDUR PENGUJIAN** | 1. Aktor menekan tombol sunting profil<br>2. Aktor diperlihatkan nama, email, password dan asuransi kesehatan yang dimiliki oleh aktor<br>3. Aktor dapat merubah salah satu data dan memasukan inputan baru<br>4. Aktor menekan tombol simpan profil |

**Tabel 7.5 Kasus *Validity Testing* Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan**

| | |
|---|---|
| **ID KASUS UJI** | **PV-005** |
| **NAMA KASUS UJI** | Pencarian rumah sakit berdasarkan asuransi kesehatan |
| **ID *USE CASE*** | FRS-104 |

**Tabel 7.5 Kasus Validity Testing Pencarian Rumah Sakit Berdasarkan Asuransi Kesehatan (lanjutan)**

| | |
|---|---|
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk melakukan pencarian rumah sakit di Kota Malang berdasarkan prefensi asuransi kesehatan |
| **PROSEDUR PENGUJIAN** | Aktor dapat memulai pencarian rumah sakit berdasarkan preferensi asuransi kesehatan yang dimiliki aktor setelah menekan menu pencarian rumah sakit berdasarkan asuransi kesehatan |

**Tabel 7.6 Kasus *Validity Testing* Pemilihan Asuransi Kesehatan**

| | |
|---|---|
| **ID KASUS UJI** | **PV-006** |
| **NAMA KASUS UJI** | Pemilihan asuransi kesehatan |
| **ID *USE CASE*** | FRS-105 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang |
| **PROSEDUR PENGUJIAN** | 5. Aktor akan ditampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang<br>6. Aktor dapat memulai pencarian asuransi kesehatan yang dimiliki oleh aktor dengan memilih dari daftar asuransi kesehatan yang telah disediakan oleh sistem<br>7. Aktor memilih asuransi kesehatan yang dimilikinya |

**Tabel 7.7 Kasus *Validity Testing* Menampilkan Lokasi Pengguna Dan Lokasi Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-007** |
| **NAMA KASUS UJI** | Menampilkan lokasi pengguna dan lokasi rumah sakit |
| **ID *USE CASE*** | FRS-106 |

**Tabel 7.7 Kasus Validity Testing Menampilkan Lokasi Pengguna Dan Lokasi Rumah Sakit (lanjutan)**

| | |
|---|---|
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima |
| **PROSEDUR PENGUJIAN** | 1. Aktor akan ditampilkan lokasi dia berada saat itu dan lokasi beberapa lokasi rumah sakit<br>2. Aktor dapat memulai pencarian rumah sakit telah disediakan oleh sistem<br>3. Aktor memilih rumah sakit |

**Tabel 7.8 Kasus *Validity Testing* Melihat Detail Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-008** |
| **NAMA KASUS UJI** | Melihat detail rumah sakit |
| **ID *USE CASE*** | FRS-107 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* |
| **PROSEDUR PENGUJIAN** | 1. Aktor akan ditampilkan nama, alamat, nomor telepon dan asuransi apa saja yang diterima di rumah sakit yang telah dipilih oleh aktor sebelumnya setelah menekan deskripsi nama dan alamat rumah sakit<br>2. Aktor menekan tombol *routing* untuk menunjukan rute dari lokasi aktor menuju ke lokasi rumah sakit |

**Tabel 7.9 Kasus *Validity Testing* Menelpon Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-009** |
| **NAMA KASUS UJI** | Menelpon rumah sakit |
| **ID *USE CASE*** | FRS-108 |

**Tabel 7.9 Kasus Validity Testing Menelpon Rumah Sakit (lanjutan)**

| | |
|---|---|
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna dapat menelpon rumah sakit |
| **PROSEDUR PENGUJIAN** | 1. Aktor akan dapat menelpon rumah sakit bila menekan nomor telpon yang tersedia di halaman detail rumah sakit<br>2. Aktor di tampilkan halaman *calling* yang mana sudah tersedia nomor telepon rumah sakit<br>3. Aktor menelpon rumah sakit |

**Tabel 7.10 Kasus *Validity Testing* Merouting Dari Lokasi Pengguna Ke Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-010** |
| **NAMA KASUS UJI** | Merouting dari lokasi pengguna ke rumah sakit |
| **ID *USE CASE*** | FRS-109 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* |
| **PROSEDUR PENGUJIAN** | 1. Aktor menekan alamat yang tersedia di halaman detail rumah sakit<br>2. Aktor akan ditampilkan halaman *API Google Maps* yang telah tersedia *routing* dari lokasi aktor menuju lokasi rumah sakit yang telah dipilih aktor sebelumnya<br>3. Aktor akan menekan tombol *start* untuk memulai *routing* dari lokasi aktor menuju lokasi rumah sakit |

**Tabel 7.11 Kasus *Validity Testing* Pencarian Rumah Sakit Seluruh Kota Malang**

| | |
|---|---|
| **ID KASUS UJI** | **PV-011** |
| **NAMA KASUS UJI** | Pencarian rumah sakit seluruh Kota Malang |
| **ID *USE CASE*** | FRS-110 |

**Tabel 7.11 Kasus Validity Testing Pencarian Rumah Sakit Seluruh Kota Malang (lanjutan)**

| | |
|---|---|
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk pencarian rumah sakit seluruh Kota Malang bagi pengguna yang tidak memiliki asuransi kesehatan |
| **PROSEDUR PENGUJIAN** | Aktor dapat memulai pencarian rumah sakit seluruh Kota Malang setelah menekan menu pencarian rumah sakit seluruh Kota Malang |

**Tabel 7.12 Kasus *Validity Testing* Melihat Info Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-012** |
| **NAMA KASUS UJI** | Melihat info rumah sakit |
| **ID *USE CASE*** | FRS-111 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menyediakan fitur bagi pengguna untuk mengetahui info dari masing-masing rumah sakit di Kota Malang |
| **PROSEDUR PENGUJIAN** | Aktor dapat memulai pencarian info rumah sakit setelah menekan menu info rumah sakit |

**Tabel 7.13 Kasus *Validity Testing* Pemilihan Info Rumah Sakit**

| | |
|---|---|
| **ID KASUS UJI** | **PV-013** |
| **NAMA KASUS UJI** | Pemilihan info rumah sakit |
| **ID *USE CASE*** | FRS-112 |
| **TUJUAN PENGUJIAN** | Untuk membuktikan bahwa sistem memiliki fungsi yang dapat digunakan untuk menampilkan daftar nama rumah sakit seluruh Kota Malang |
| **PROSEDUR PENGUJIAN** | 1. Aktor akan ditampilkan daftar seluruh nama rumah sakit seluruh Kota Malang<br>2. Aktor dapat memulai pencarian rumah sakit yang ingin diketahui infonya dengan memilih dari daftar rumah sakit yang telah disediakan oleh system<br>3. Aktor memilih rumah sakit yang ingin diketahui infonya |

Setelah didefinisikan kasus uji, berikutnya adalah proses penentuan hasil pengujian, dimana akan menunjukan hasil yang diperoleh dari pengujian beserta status pengujiannya yang dapat dilihat pada Tabel 7.14.

**Tabel 7.14 Hasil *Validity Testing***

| ID | Nama Kasus Uji | Hasil Yang Diharapkan | Hasil Yang Diperoleh | Status |
|---|---|---|---|---|
| **PV-001** | *Login* | Sistem menyediakan fitur bagi pengguna untuk melakukan autentifikasi *Login* dengan menginputkan *email* dan *password* | Sistem menyediakan fitur bagi pengguna untuk melakukan autentifikasi *Login* dengan menginputkan *email* dan *password* | Valid |
| **PV-002** | *Register* | Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | Valid |
| **PV-003** | *View* | Sistem menyediakan fitur bagi pengguna untuk melihat data diri yang berisikan nama, *email, password*, dan asuransi kesehatan yang dimiliki | Sistem menyediakan fitur bagi pengguna untuk melihat data diri yang berisikan nama, *email, password*, dan asuransi kesehatan yang dimiliki | Valid |
| **PV-004** | *Edit* | Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | Sistem menyediakan fitur bagi pengguna untuk *Register* dirinya agar dapat melakukan *Login* dengan menginputkan nama, *email*, *password*, dan asuransi kesehatan yang dimiliki | Valid |

**Tabel 7.14 Hasil Validity Testing (lanjutan)**

| | | | | |
|---|---|---|---|---|
| **PV-005** | Pencarian rumah sakit berdasarkan asuransi kesehatan | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit di Kota Malang berdasarkan prefensi asuransi kesehatan yang dimiliki oleh pengguna | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit di Kota Malang berdasarkan prefensi asuransi kesehatan yang dimiliki oleh pengguna | Valid |
| **PV-006** | Pemilihan asuransi kesehatan | Sistem menampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang | Sistem menampilkan daftar seluruh nama asuransi kesehatan yang mana asuransi tersebut telah diterima oleh beberapa rumah sakit di Kota Malang | Valid |
| **PV-007** | Menampilkan lokasi pengguna dan lokasi rumah sakit | Sistem menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima asuransi kesehatan dalam bentuk *Maps* | Sistem menampilkan lokasi pengguna dan lokasi rumah sakit yang menerima asuransi kesehatan dalam bentuk *Maps* | Valid |
| **PV-008** | Melihat detail rumah sakit | Sistem menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* | Sistem menampilkan deskripsi nama dan alamat rumah sakit dalam bentuk *Maps* kemudian menampilkan tombol *routing* | Valid |
| **PV-009** | Menelpon rumah sakit | Sistem menyediakan fitur bagi pengguna dapat menelpon rumah sakit | Sistem menyediakan fitur bagi pengguna dapat menelpon rumah sakit | Valid |
| **PV-010** | Merouting dari lokasi pengguna ke rumah sakit | Sistem menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* | Sistem menampilkan *routing* dari lokasi pengguna ke lokasi rumah sakit yang telah dipilih dengan *API Google Maps* | Valid |

**Tabel 7.14 Hasil Validity Testing (lanjutan)**

| | | | | |
|---|---|---|---|---|
| **PV-011** | Pencarian rumah sakit seluruh Kota Malang | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit seluruh Kota Malang bagi pengguna yang tidak memiliki asuransi kesehatan | Sistem menyediakan fitur bagi pengguna untuk pencarian rumah sakit seluruh Kota Malang bagi pengguna yang tidak memiliki asuransi kesehatan | Valid |
| **PV-012** | Melihat info rumah sakit | Sistem menyediakan fitur bagi pengguna untuk mengetahui info dari masing-masing rumah sakit di Kota Malang | Sistem menyediakan fitur bagi pengguna untuk mengetahui info dari masing-masing rumah sakit di Kota Malang | Valid |
| **PV-013** | Pemilihan info rumah sakit | Sistem menampilkan daftar nama rumah sakit seluruh Kota Malang | Sistem menampilkan daftar nama rumah sakit seluruh Kota Malang | Valid |

### *7.1.2 Usability Testing*

*Usability testing* ditujukan untuk mengkaji seberapa puas dan mudah aplikasi yang dibangun oleh pengguna baik dari kalangan pengguna personal/masyarakat umum. Pengujian ini dilakukan dengan cara menyebarkan kuisioner dan memberikan task kepada calon pengguna yakni masyarakat umum yang mana dia memiliki asuransi kesehatan yang ditemukan di sekitar pemukiman warga dan jumlah kuisioner berjumlah 20 yang diperkuat oleh Sauro (2013), dikarenakan indikator *usability* yang lebih baik adalah dengan menggunakan 20 responden. Dokumentasi dari *usability testing* dapat ditemukan dalam bagian lampiran (Lampiran C).

Kuisioner ditargetkan kepada 20 orang responden dengan status pengguna personal yang dalam pengujian aplikasi dicoba responden. Responden yang mengisi kuisioner sebelumnya harus terbiasa menggunakan perangkat Android. Hasil pengisian kuisioner oleh responden ditunjukan pada Tabel 7.15.

**Tabel 7.15 Hasil Rekapitulasi Kuisioner *Usability Testing***

| **NO** | **PERNYATAAN** | **JAWABAN** | | | | | **Total** |
|---|---|---|---|---|---|---|---|
| | | **STS** | **TS** | **N** | **S** | **SS** | |
| | | **1** | **2** | **3** | **4** | **5** | |
| *Usefullness* | | | | | | | |

**Tabel 7.15 Hasil Rekapitulasi Kuisioner Usability Testing (lanjutan)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | Aplikasi ini membantu mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 7 | 13 | 20 |
| 2 | Aplikasi ini membuat saya menghemat waktu saat hendak mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 6 | 14 | 20 |
| 3 | Aplikasi ini sangat berguna bagi saya | 0 | 0 | 5 | 5 | 10 | 20 |
| 4 | Aplikasi ini mempermudah saya mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 6 | 14 | 20 |
| 5 | Aplikasi ini sesuai dengan apa yang saya harapkan. | 0 | 0 | 0 | 9 | 11 | 20 |
| *Easy to Learn* | | | | | | | |
| 6 | Saya dapat mempelajari menggunakan aplikasi ini dengan cepat. | 0 | 0 | 0 | 8 | 12 | 20 |
| 7 | Saya mudah mengingat bagaimana menggunakan aplikasi ini. | 0 | 0 | 0 | 6 | 14 | 20 |
| 8 | Saya dengan mudah menggunakan aplikasi ini. | 0 | 0 | 0 | 8 | 12 | 20 |
| 9 | Saya dengan cepat menguasai aplikasi ini. | 0 | 0 | 0 | 5 | 15 | 20 |
| *Easy to Use* | | | | | | | |
| 10 | Aplikasi ini mudah digunakan. | 0 | 0 | 0 | 6 | 14 | 20 |
| 11 | Aplikasi ini dapat digunakan dengan sederhana. | 0 | 0 | 3 | 5 | 12 | 20 |
| 12 | Aplikasi ini dapat digunakan oleh semua kalangan masyarakat. | 0 | 0 | 1 | 5 | 14 | 20 |
| 13 | Aplikasi ini memerlukan langkah-langkah sederhana untuk dapat mencapai apa yang saya inginkan. | 0 | 0 | 2 | 5 | 13 | 20 |

**Tabel 7.15 Hasil Rekapitulasi Kuisioner Usability Testing (lanjutan)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 | Aplikasi ini dapat digunakan secara fleksibel. | 0 | 0 | 1 | 9 | 10 | 20 |
| 15 | Tidak membutuhkan banyak usaha untuk menggunakan aplikasi ini. | 0 | 0 | 3 | 7 | 10 | 20 |
| 16 | Saya dapat menggunakan aplikasi ini tanpa instruksi tertulis. | 0 | 0 | 0 | 6 | 14 | 20 |
| 17 | Saya dapat dengan cepat dan mudah memperbaiki kesalahan saya dalam menggunakan aplikasi. | 0 | 0 | 0 | 7 | 13 | 20 |
| 18 | Saya dapat menggunakan aplikasi ini dengan praktis saat membutuhkannya. | 0 | 0 | 0 | 3 | 17 | 20 |
| *Satisfaction* | | | | | | | |
| 19 | Saya puas dengan aplikasi ini. | 0 | 0 | 0 | 4 | 16 | 20 |
| 20 | Saya akan merekomendasikan aplikasi ini kepada orang lain. | 0 | 0 | 0 | 3 | 17 | 20 |
| 21 | Aplikasi ini menyenangkan untuk digunakan. | 0 | 0 | 0 | 4 | 16 | 20 |
| 22 | Aplikasi ini bekerja seperti yang saya inginkan. | 0 | 0 | 0 | 5 | 15 | 20 |
| 23 | Aplikasi ini sangat bagus. | 0 | 0 | 0 | 6 | 14 | 20 |
| 24 | Saya ingin memiliki aplikasi ini. | 0 | 0 | 0 | 7 | 13 | 20 |
| 25 | Aplikasi ini nyaman untuk digunakan. | 0 | 0 | 0 | 5 | 15 | 20 |

## 7.2 Analisis

Analisis dilakukan terhadap tiap pengujian yang dilakukan yakni analisis *validity testing* dan anilisis *usability testing*. Analisis dilakukan untuk mempermudah mendapatkan sebuah kesimpulan dari penelitian ini.

### 7.2.1 Analisis Hasil *Validity Testing*

Proses analisis pada hasil *validity testing* dilakukan dengan membandingkan kesesuai hasil uji dengan hasil yang diharapkan saat merancang aplikasi. Bila hasil uji sesuai dengan perancangan maka aplikasi tersebut adalah valid atau memenuhi kebutuhan fungsional. Bila tidak sesuai dengan hasil yang diharapkan dari perancangan aplikasi maka tidak valid atau tidak memenuhi kebutuhan fungsional. Berdasarkan hasil pengujian dapat disimpulkan bahwa implementasi pengembangan aplikasi *mobile* untuk pencarian rumah sakit berdasarkan preferensi asuransi kesehatan telah nenebuhi analisis dan perancangan sistem dikarenakan semua fitur yang diuji adalah valid dengan tingkat validitas mencapai 100%.

### 7.2.2 Analisis Hasil *Usability Testing*

Setelah dilakukan *usability testing* kepada 20 responden, hasil yang didapatkan adalah berupa suatu akumulasi poin terhadap tiap aspek yang ditunjukan pada kepala dimana poin 1 berarti sangat tidak setuju, 2 berarti tidak setuju, 3 berarti netral, 4 berarti setuju, 5 berarti sangat setuju. Aplikasi akan dinyatakan memenuhi indeks dari *usability testing* bila rata-rata persentase dari tiap kriteria melebihi 60% sesuai dengan interpretasi skor yang ditunjukan pada. Interprestasi dari *usability testing* ditunjukan pada Tabel 7.16 dan hasil dari *usability testing* ditunjukan pada Tabel 7.17 dan hasil status dari *usability testing* ditunjukan pada Tabel 7.18.

**Tabel 7.16 Interpretasi Skor *Likert***

| Skor *Likert* | Interpretasi skor dengan interval = 20 | Pilihan |
|---|---|---|
| 1 | 0% - 19,99% | Sangat tidak setuju |
| 2 | 20% - 39,99% | Tidak setuju |
| 3 | 40% - 59,99% | Netral |
| 4 | 60% - 79,99% | Setuju |
| 5 | 80% - 100% | Sangat setuju |

Keterangan:

Interval = 20 didapatkan dari pembagian jumlah skor *Likert* dengan nilai 100.

**Tabel 7.17 Hasil Perhitungan *Usability Testing***

| NO | PERNYATAAN | JAWABAN | | | | | Total | Indeks |
|---|---|---|---|---|---|---|---|---|
| | | STS | TS | N | S | SS | | |
| | | 1 | 2 | 3 | 4 | 5 | | |
| *Usefullness* | | | | | | | | |

**Tabel 7.17 Hasil Perhitungan Usability Testing (lanjutan)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | Aplikasi ini membantu mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 7 | 13 | 20 | 93 |
| 2 | Aplikasi ini membuat saya menghemat waktu saat hendak mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 6 | 14 | 20 | 94 |
| 3 | Aplikasi ini sangat berguna bagi saya | 0 | 0 | 5 | 5 | 10 | 20 | 85 |
| 4 | Aplikasi ini mempermudah saya mencari rumah sakit berdasarkan asuransi kesehatan | 0 | 0 | 0 | 6 | 14 | 20 | 94 |
| 5 | Aplikasi ini sesuai dengan apa yang saya harapkan. | 0 | 0 | 0 | 9 | 11 | 20 | 91 |
| **Rata-rata *Usefullness*** | | | | | | | | 91,4 |
| *Easy to Learn* | | | | | | | | |
| 6 | Saya dapat mempelajari menggunakan aplikasi ini dengan cepat. | 0 | 0 | 0 | 8 | 12 | 20 | 92 |
| 7 | Saya mudah mengingat bagaimana menggunakan aplikasi ini. | 0 | 0 | 0 | 6 | 14 | 20 | 94 |
| 8 | Saya dengan mudah menggunakan aplikasi ini. | 0 | 0 | 0 | 8 | 12 | 20 | 92 |
| 9 | Saya dengan cepat menguasai aplikasi ini. | 0 | 0 | 0 | 5 | 15 | 20 | 95 |
| **Rata-rata *Easy to Learn*** | | | | | | | | 93,25 |
| *Easy to Use* | | | | | | | | |
| 10 | Aplikasi ini mudah digunakan. | 0 | 0 | 0 | 6 | 14 | 20 | 94 |

**Tabel 7.17 Hasil Perhitungan Usability Testing (lanjutan)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | Aplikasi ini dapat digunakan dengan sederhana. | 0 | 0 | 3 | 5 | 12 | 20 | 89 |
| 12 | Aplikasi ini dapat digunakan oleh semua kalangan masyarakat. | 0 | 0 | 1 | 5 | 14 | 20 | 93 |
| 13 | Aplikasi ini memerlukan langkah-langkah sederhana untuk dapat mencapai apa yang saya inginkan. | 0 | 0 | 2 | 5 | 13 | 20 | 91 |
| 14 | Aplikasi ini dapat digunakan secara fleksibel. | 0 | 0 | 1 | 9 | 10 | 20 | 89 |
| 15 | Tidak membutuhkan banyak usaha untuk menggunakan aplikasi ini. | 0 | 0 | 3 | 7 | 10 | 20 | 87 |
| 16 | Saya dapat menggunakan aplikasi ini tanpa instruksi tertulis. | 0 | 0 | 0 | 6 | 14 | 20 | 94 |
| 17 | Saya dapat dengan cepat dan mudah memperbaiki kesalahan saya dalam menggunakan aplikasi. | 0 | 0 | 0 | 7 | 13 | 20 | 93 |
| 18 | Saya dapat menggunakan aplikasi ini dengan praktis saat membutuhkannya. | 0 | 0 | 0 | 3 | 17 | 20 | 97 |
| **Rata-rata *Easy to Use*** | | | | | | | | 91,9 |
| *Satisfaction* | | | | | | | | |
| 19 | Saya puas dengan aplikasi ini. | 0 | 0 | 0 | 4 | 16 | 20 | 96 |
| 20 | Saya akan merekomendasikan aplikasi ini kepada orang lain. | 0 | 0 | 0 | 3 | 17 | 20 | 97 |
| 21 | Aplikasi ini menyenangkan untuk digunakan. | 0 | 0 | 0 | 4 | 16 | 20 | 96 |

**Tabel 7.17 Hasil Perhitungan Usability Testing (lanjutan)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 22 | Aplikasi ini bekerja seperti yang saya inginkan. | 0 | 0 | 0 | 5 | 15 | 20 | 95 |
| 23 | Aplikasi ini sangat bagus. | 0 | 0 | 0 | 6 | 14 | 20 | 94 |
| 24 | Saya ingin memiliki aplikasi ini. | 0 | 0 | 0 | 7 | 13 | 20 | 93 |
| 25 | Aplikasi ini nyaman untuk digunakan. | 0 | 0 | 0 | 5 | 15 | 20 | 95 |
| **Rata-rata *Satisfaction*** | | | | | | | | 95,1 |

**Tabel 7.18 Hasil Status *Usability Testing***

| Aspek Penilaian | Rata-rata Persentase (%) | Status |
|---|---|---|
| *Usefulness* | 91,4 | Sangat setuju |
| *Ease to learn* | 93,25 | Sangat setuju |
| *Ease to use* | 91,9 | Sangat setuju |
| *Satisfaction* | 95,1 | Sangat setuju |
| **Rata-rata** | | **92,91 %** |

Dari *usability testing* yang dilakukan menunjukan bahwa aplikasi memenuhi kriteria yang ada dengan rata-rata persentase semua kriteria sebesar 92,91%, sehingga menunjukkan bahwa aplikasi pencarian rumah sakit di Kota Malang berdasarkan asuransi kesehatan memenuhi kriteria *usability testing*. Dari sisi *usefulness* didapatkan *score* 91,4% yang dapat disimpulkan bahwa aplikasi *Hospital Locator* sangat berguna dan memudahkan pengguna untuk mencari rumah sakit berdasarkan asuransi kesehatan. Dari sisi *easy to learn* dan *easy to use* mendapatkan *score* 93,25% dan 91,9% yang dapat disimpulkan bahwa aplikasi *Hospital Locator* adalah aplikasi yang mudah digunakan oleh pengguna. Dari sisi *satisfication* mendapatkan *score* 95,1% yang dapat disimpulkan bahwa pengguna merasa puas dengan adanya aplikasi *Hospital Locator* yang dapat mencari rumah sakit berdasarkan asuransi kesehatan yang dimiliki oleh pengguna.

# BAB 8 PENUTUP

Bab ini berisikan kesimpulan dari hasil penelitian dan saran bagi pembaca. Kesimpulan berisi analisis hasil pengujian untuk menjawab rumusan masalah yang telah dijelaskan pada tahap pendahuluan. Sedangkan saran ditujukan bagi pembaca yang ingin melanjutkan penelitian ke tahap selanjutnya agar menjadi lebih baik.

## 8.1 Kesimpulan

Kesimpulan dari penelitian ini adalah:

1. Aplikasi dirancang menggunakan pendekatan *Object Oriented* dengan pemodelan kebutuhan menggunakan *use case diagram* dan *use case scenar*io serta pemodelan perancangan menggunakan pendekatan *UX*, *sequence diagram* dan *class diagram*. Kemudian di implementasikan dengan menggunakan bahasa pemrograman Java IDE Android *Studio* 2.2.3 serta menggunakan *DBMS MySQL* untuk mengimplementasikan web service/ basis data aplikasi.
2. Pengujian dilakukan dengan dua tahapan yaitu dengan *validity testing* dan *usability testing*. Berdasarkan hasil *validity testing* pada aplikasi menunjukan nilai dengan persentase 100% yang berarti sistem sudah memenuhi kebutuhan fungsional. Sedangkan untuk tingkat *usability* dari aplikasi berdasarkan aspek *usefulness, ease to learn, ease to use* dan *satisfaction* menunjukan rata-rata sebesar 92,91% yang berarti aplikasi telah terbukti mempermudah pengguna untuk mencari dan memberikan pertolongan saat terjadi permasalahan dengan mencari rumah sakit yang menerima asuransi kesehatan.

## 8.2 Saran

Berdasarkan kesimpulan tersebut, saran dari penulis adalah:

1. Disarankan untuk kedepannya dikembangkan kembali dengan bahasa pemrograman IOS yang mana mungkin dapat menarik lebih banyak pengguna.
2. Disarankan untuk kedepannya cakupan yang dijangkau lebih luas dari pada Kota Malang, seperti kabupaten Malang, Batu, dll.
3. Disarankan untuk kedepannya menambahkan objek instansi kesehatan lainnya seperti klinik, rumah bersalin atau lab untuk pengecekan kesehatan.
4. Disarankan untuk kedepannya agar mengelompokan pencarian rumah sakit yang menerima asuransi kesehatan berdasarkan jenis penyakit yang dapat di klaim di rumah sakit, agar lebih spesifik.

# DAFTAR PUSTAKA


Aelani, K. & Falahah. 2012. *Pengukuran Usability Sistem Menggunakan USE Questionnaire.*

Abran, A. & Moore, WJ. (2004). *Guide to the Software Engineering Body of Knowledge.*

Ambler, SW. 2012. *Disciplined Agile Delivery (DAD): The Foundation for Scaling Agile.*

Brodkin, J. 2012. 5 things we love about Android.

Cockburn , A. 2002. Use cases, ten years later.

Departemen Kesehatan RI. 1999. *Pedoman Upaya Peningkatan Mutu Pelayanan Rumah Sakit*. Jakarta.

Elgin, B. 2005. Google Buys Android for Its Mobile Arsenal.

Gemino, A. & Parker, D. 2009. Use case diagrams in support of use case modeling: Deriving understanding from the picture.

Hartantyo, TP. 2014. *Rancangbangun Pencarian Lokasi Rumah Sakit Dan Puskesmas di Wilayah Tegal Berbasis Android*.

Ingraham, N. .2012. Google Maps for Android.

Mahsun. 2011. Metode Penelitian Bahasa.

McCann, J. (2012). Android 4.1 Jelly Bean source code released.

Nafsiah, S. 2000. *Prof. Hembing pemenang the Star of Asia Award: pertama di Asia ketiga di dunia.*

Nielsen. 2012. *Usability 101: Introduction.* [Online] Available at: (http://www.useit.com/alertbox/20030825.html) [Accessed 10 12 2015].

OMG. 2011. OMG Unified Modeling Language, Superstructure, V2.4.1.

Pressman. 2001. *Software Engineering A Practitioner's Approach. 5th ed. New York: McGraw-Hill Higher Education.*

Putro, A., Tolle, H., & Kharisma, A. 2017. *Rancang Bangun Aplikasi Penawaran dan Pencarian Kerja Paruh Waktu (Part Time) Berbasis Lokasi.*

Risnita. 2014. Pengembangan Skala Model Likert. *Pengembangan Skala Model Likert,* Volume 3, pp. 1-14.

Sauro, J. 2013. *10 THINGS TO KNOW ABOUT THE SYSTEM USABILITY SCALE (SUS).*

Simamarta, J. 2010. Rekayasa Perangkat Lunak.

Sparks, G. 2011. Pemodelan Database di UML.

Thabrany, H. 2001, Asuransi Kesehatan di Indonesia. Pusat Kajian Ekonomi Kesehatan

Wiley, J. 2005. Location-based Services Fundamentals and Operation.